METHOD & SYSTEM FOR INFORMATION COMMUNICATION BETWEEN POTENTIAL POSITIONEES AND POSITIONORS

Background of the Invention

One objective of job placement services is to be a tool for employers to utilize in finding qualified job applicants or job seekers. Employment services are now available online, such as on the Internet, and have become a key component of job searching, placement, and for assisting employers to find job seekers.

Current job placement online systems are not sufficiently user-friendly, do not contain the necessary flexibility needed to track skills, and do not manage employer data. Also, current systems do not sufficiently track other employee data, or the interaction between job seekers and potential employers. One such current system is United States Patent No. 5,832,497, currently implemented at http://www.monster.com on the Internet. Another such current system is located at http://www.ohioworks.com on the Internet. The present invention is provided to solve these and other problems.

Summary of the Invention

According to a broad aspect of a preferred embodiment of the invention, a method of matching a potential positionee and a potential positionor by providing the potential positionee with a positionee information entry interface for electronically entering positionee information comprising the potential positionee's actual qualifications, the positionee information being stored in a database is provided. Matching is further accomplished by providing the potential

positionor with a positionor information entry interface for electronically entering positionor information comprising at least one target qualification for a position, the positionor information being stored in the database. Matching is based on determining whether the positionee information correlates with the positionor information. The method then provides for the creating of a correlated information list of correlated information and presenting the correlated information for review.

The positionee information may be maintained as confidential and may also include contact information for receiving communication, veteran information, transportation information for position site availability, work history information, and education information. In addition, the positionee information may also include at least one position category and actual qualifications of at least one skill relating to the position category. The qualifications are selected from a positionee skills listing. Positionor information also includes positionor entity information. The method also verifies the existence of the potential positionor using the positionor entity information.

Positionor information may include positionor contact information, as well as a plurality of target qualifications for the position, salary information required for the position, benefits information for the position, site location information for the position, special programs participation information, and a position category. The position category contains at least one skill required for the position. The position category may also include at least one skill that would be nice to have, but that is not required.

The target qualifications reflect at least one skill selected from a positionor skills listing and may include at least one skill selected from a positionee skills listing. The target

qualifications are useful in determining whether the positionee information correlates with the positionor information. The resulting correlated information contains only potential positionees or potential positionors for which a correlation has taken place. Positionees and/or positionors may select one or more skills from a skills listing to identify actual qualifications or target qualifications. Particular skills can be added and/or deleted to/from the skills listing. Furthermore, the positionee information and/or the positionor information can be edited, and if editing occurs correlation is determined again.

The correlated information is rank-ordered according to one or more of the following ranking criteria: skills that would be nice to have, but not required for the position; special programs information; and, veteran information. The correlated information list may be used as a trial correlated information list including only the number of correlated potential positionees for a potenial positionor, without an identification of the potential positionees. Then an order for a position may be submitted.

The correlated information list includes a list of correlated potential positionors for consideration by one of the potential positionees, and a list of correlated potential positionees for consideration by one of the potential positionors. The potential positionee can choose to be removed from the correlated information list from which the potential positionor considers such potential positionee.

At least one step is performed over a computer network, such as a LAN or the Internet. The positionee and positionor information can be inputted over a computer network, such as a LAN or the Internet. The correlated information may also provided over a computer network, such as a LAN or the Internet via e-mail, phone, fax, or letter.

The method of matching also uses a preexisting selection hierarchy with the steps of selecting a position from a preexisting set of positions; and selecting a skill from a preexisting set of skills relating to the selected position. The preexisting set of positions relate to the selected field from the preexisting set of fields. In addition, the preexisting selection hierarchy includes preexisting sets of positions, with each preexisting set of positions relating to one field within the preexisting set of fields. The preexisting selection hierarchy may also include preexisting sets of skills, with each preexisting set of skills relating to one position within the preexisting set of positions. A skill may also be displayed under multiple preexisting sets of positions in order to facilitate matching across job titles or job categories. Fields, skills, and positions can be added or deleted. Additionally, the preexisting selection hierarchy is stored in electronically readable memory.

A computer program for matching a potential positionee and a potential positionor is also provided. The computer program includes a code segment providing the potential positionee with a positionee information entry interface for electronically entering positionee information comprising the potential positionee's actual qualifications, the positionee information being stored in a database; a code segment providing the potential positionor with a positionor information entry interface for electronically entering positionor information comprising at least one target qualification for a position, the positionor information being stored in the database; a code segment for determining whether the positionee information correlates with the positionor information; a code segment creating a correlated information list of correlated information; and a code segment providing the correlated information for review.

The method further provides for participation in assisted position placement within

special programs. The potential positionee utilizes a positionee information entry interface for electronically entering positionee information for determining whether the potential positionee qualifies for a special program. The positionee information is then stored in a database. In addition, the potential positionor utilizes a positionor information entry interface for electronically entering positionor information to determine whether the potential positionor is participating in one or more special programs, including: DOC 7-B; MANG; TANF; WOTC; HTF; NAFS; Title I; International Registry: Sr. Comm. Service Employment Program; and Title II. The positionor information is then stored in the database as well. The method further determines whether the positionee information correlates with the positionor information, thereby creating a correlated information list of correlated information. This correlated information list provides the correlated information for review.

Brief Description of the Figures

FIGURE 1 is an overview of the system

FIGURE 2 is a web page illustration of job seeker registration

FIGURE 3 is a diagram representing skill selection for a job seeker

FIGURE 4 is a web page illustration of employer registration

FIGURE 5 is a representation of the general organization of web pages to provide a customized menu of function options

FIGURE 6 is an illustration of the general layout of a typical web page of the system

FIGURE 7 is a web page illustration of the home/login page

FIGURE 8 is a web page illustration of the job seeker menu page

FIGURE 9 is a web page illustration of the employer menu page

FIGURE 10 is a web page illustration of the staff menu page

FIGURE 11 is a web page illustration of the job seeker search page

FIGURE 12 is a web page illustration of a list page

FIGURE 13 is a web page illustration of a detail page

FIGURE 14 is a web page illustration of a job seeker registration page

FIGURE 15 is a web page illustration of the forms for job seeker registration

FIGURE 16 is a web page illustration of the veteran information

FIGURE 17 is a web page illustration of the transportation form and work information form

FIGURE 18 is a web page illustration of the optional forms for work history

FIGURE 19 is a web page illustration of the optional forms for education

FIGURE 20 is a web page illustration of a job position

FIGURE 21 is a web page illustration of a skills checklist

FIGURE 22 is a web page illustration of the employer login

FIGURE 23 is a web page illustration of the company information and contact information forms

FIGURE 24 is a web page illustration of a job order worksheet

FIGURE 25 is a web page illustration of a worksite information form

FIGURE 26 is a web page illustration of the ability to hide/reveal employer contact information

FIGURE 27 is an example of a web page illustration of skills for a job order

FIGURE 28 is an example of web page illustration of the experience level for skills

FIGURE 29 is a web page illustration of a qualified candidate list

FIGURE 30 is an example of a web page illustration of recruiting actions

FIGURE 31 is a web page illustration of a staff menu

FIGURE 32 is a web page illustration of a special programs page

FIGURE 33 is an example of a web page illustration of a staff search screen

FIGURE 34 is an example of a web page illustration of a staff search screen for job seekers

FIGURE 35 is an example of a web page illustration of a staff edit screen for editing employer company information and employer contact information

FIGURE 36 is an example of a web page illustration of a staff edit screen for editing transportation information and work information

FIGURE 37 is an example of a web page illustration of rank-ordering of matches

FIGURE 38 is an illustration of the invention's multi-tier technical architecture

FIGURE 39 illustrates the integration of components in the potential positionee/positionor system

FIGURE 40 shows transaction flow in the potential positionee/positionor system

FIGURE 41 illustrates how the HTTP Router interacts with other components within the potential positionee/positionor technical environment

FIGURE 42 illustrates the flow of information between the web servers and application servers

FIGURE 43 illustrates the interaction between the application servers and the other

system components

FIGURE 44 illustrates the interaction between the database servers and the other system components

FIGURE 45 illustrates the major components involved in a highly available database design

FIGURE 46 illustrates the major components required to achieve parallelism in a database design

FIGURE 47 illustrates interaction between the three "subnetworks" of the invention's network environment

FIGURE 48 is a diagram of the web servers and the components they communicate with

FIGURE 49 shows the application servers' architecture and interconnection to other components

FIGURE 50 illustrates the use of a database cluster

FIGURE 51 summarizes the Sun Cluster, mirrored disks, and DB2 UDB EEE's interaction with these components

FIGURE 52 illustrates at a high-level how the Servlets, JSPs and Extensions work within a iAS server and the points of interaction with the web server, database servers, and other external services

FIGURE 53 illustrates the tiers of a web application and which iAS and other components address which tier

FIGURE 54 shows the flow of a iAS based application

FIGURE 55 illustrates the division of the network architecture into zones where traffic flow can be denied for security purposes

FIGURE 56 is a more specific diagram of the zone division for security purposes

FIGURE 57 is a diagram of the potential positionee/positionor's database design showing the top-left portion of an overall diagram

FIGURE 58 is a continuation of the diagram of FIGURE 57 showing the top-right portion of an overall diagram

FIGURE 59 is a continuation of the diagram of FIGURE 57 showing the bottom-left portion of an overall diagram

FIGURE 60 is a continuation of the diagram of FIGURE 57 showing the bottom-right portion of an overall diagram

Detailed Description

The potential positionee/positionor system of the present invention provides employers with the best qualified candidates available by matching the skills needed by the employers to the skills held by job seekers. The system delivers the following benefits:

Emphasizing customer choice and self-service options to employers and job-seekers;

Developing an employer database that can track job order activity, success rates, and employer preferences;

Providing a flexible skills repository that can grow and change with business needs;

Allowing access to the system through a network;

Opening the system to a broader field of job candidates that may not otherwise be included in

the labor exchange; and

Enabling continuous improvement.

This potential positionee/positionor system provides the employment service organizations with a tool to improve customer service, raises the image of such organizations with the employer community and job seeker community, and leverages the organization's staff to provide specialized labor market services to employers and job seekers.

The potential positionee/positionor system provides a method for matching employer's job openings with job seekers based on matching of standardized skills. Employers enter job orders and describe the required skills for the position. Applicants record the skills acquired through their various experiences and employment situations. The potential positionee/positionor system determines which job seekers match with which job openings. Based on restrictions specified by either party, the potential positionee/positionor system can then notify the parties of matches.

Employers and job seekers may use the potential positionee/positionor system through a secure and robust self service application via a network such as the Internet using a standard interface -- a standard web browser. The Employers and job seekers may also work directly with the staff of an organization, such as an employment placement firm or an employment security component of a state government.

Organizational staff may also access the staff related functions of the potential positionee/positionor system by using standard interfaces, such as web browsers accessing the potential positionee/positionor system directly from the organization's LAN/WAN or via the Internet.

The potential positionee/positionor system is a robust business application with several subsystems, a relational database, with high availability and performance requirements and interfaces to legacy applications. The potential positionee/positionor system is both a mission critical business application and also a self-service Internet application for employers and job seekers. In the present form of the invention, the potential positionee/positionor system must support the following users:

Employers;

Job Seekers; and

Employment Organization Staff.

The application and technical infrastructure architectures of the potential positionee/positionor system are highly scalable and able to accommodate dramatic increases in the transaction volume without requiring a re-design of the application. The technical infrastructure is able to scale both vertically (within a server by adding CPU, memory, disk, etc.) and horizontally (by adding additional servers).

The technical architecture is implemented using a multi-tiered architecture. As mentioned, user access to the system can be provided through standard web browsers running on a Windows™ based environment on a personal computer. Other platforms are possible, as one of ordinary skill would understand. External users can access the system via the Internet while internal users can access the system via a network. The web browsers communicate with a collection of web servers. These web servers can serve both static content such as static pages and images. Requests for dynamically generated pages can be forwarded to application servers. The dynamic content can be generated via programs elicited by the application servers. These

programs can access and manipulate persistent data via a DB2 database on a separate server.

Some of the various application protocols which can be used for communication between the components of the potential positionee/positionor system include:

FTP (File Transfer Protocol) – A cross-platform protocol for transferring files to and from computers on the Internet. .

HTTP (HyperText Transport Protocol) – An Internet protocol providing a means for Web clients and servers to communicate with one another, primarily through the exchange of requests from the client and responses from the server.

HTTPS (Secure HTTP) – This is a secure version of the HTTP protocol described above. It is implemented as HTTP within an SSL session.

HTML (HyperText Markup Language) – HTML is not a protocol. It is a markup language that describes the structure of a Web document's content plus some behavioral characteristics. All Web browsers are able to understand and interpret this standard language resulting in a cross-platform mechanism for transmitting formatted "screens" and forms. There are several versions of HTML in wide use – ranging from HTML v1.1 to v3.2. The differences in these version are in their support of advanced HTML tags and features including tables, forms, frames, style sheets, layers, font specification, and scripting languages.

NCP/KCP (Netscape/Kiva Communication Protocol) – This is a proprietary protocol used between iPlanet Application Servers to perform data synchronization, exchange performance information, and implement fail-over and load-balancing.

Net.Data – Used for transferring database data manipulation transaction between the Application servers and the database server.

SMTP (Simple Mail Transport Protocol) – A standard protocol used to send electronic mail messages.

SNMP (Simple Network Management Protocol) – A standard protocol used to monitor hosts, routers, and the networks to which they are attached.

SSL (Secure Sockets Layer) – A protocol used to conduct secure encrypted transmission sessions between clients and servers to ensure the privacy of the information in the transmissions as it travels through the network. The other application layer protocols can use SSL to allow an encrypted form of the application layer protocol. For example, the HTTPS protocol is HTTP transmissions within an SSL session.

TCP/IP (Transmission Control Protocol / Internet Protocol) – The standard transport level protocol suite that provides the reliable, full duplex, stream service on which many application protocols depend.

Telnet (terminal access) – The standard application protocol that provides remote login and terminal access to the servers from the network.

The entire potential positionee/positionor application can be run from a web browser. All web page content can be delivered to the web browsers by the two web servers. It should be understood that one, two, or more servers can be used to implement the present invention. Static (non-changing) content can be hosted and served directly by the web servers. In the case of requests for content to be generated dynamically, the web servers can pass the request through to the application servers and can pass the response back through to the web browsers.

Although the user interface of the potential positionee/positionor system can be through a web browser, the potential positionee/positionor system is a robust business application. The

application logic is implemented on the application servers, and Database access requests can be initiated by the application servers.

In one form of the present invention, the database servers provide the functionality for the data access layer of the application. Batch processing application components are co-hosted with the database services on the same physical servers. The application servers and the batch processing modules are the "clients" to the database services. The batch processing modules exchange information with legacy mainframe applications via periodic interface exports and imports. For the database to be highly available, single points of failure must be reduced. This requires that there be at least two systems combined into a database "cluster". A database cluster is a set of two or more servers that act in cooperation to process data requests. In addition to having two physical servers, there must be fault tolerance built into the disk subsystem. The database cluster is able to survive a disk failure. Connectivity between the cluster members is redundant to reduce the cluster communications path as a single point of failure.

The design for the potential positionee/positionor system may require the use of two physical servers for high availability. There are several ways of achieving parallelism in a database. In one form of the present invention, a parallel database is a group of two or more database servers dividing the work of presenting a single logical database to the client.

The entire network infrastructure for the potential positionee/positionor system can use the TCP/IP protocol. The potential positionee/positionor system users can access the system using a web browser utilizing the HTTP protocol. Internal communications between the servers is TCP/IP based. Administration and management can be performed via HTTP, TELNET, and SNMP.

In one form of the present invention, due to the extensive use of TCP/IP, the configuration of Netscape browsers is changed so that packets do not need to be converted between IP and IPX. The current IPX/IP gateway can be reconfigured to perform a web proxy function only. The proxy can provide access to the Internet. Packets addressed to the potential positionee/positionor system from the LAN or WAN will go directly to the LocalDirector and on to the potential positionee/positionor system web servers. This reconfiguration will increase performance, and eliminate a potential bottleneck and point of failure. In addition to potential positionee/positionor system access, Internet access performance will be improved by this reconfiguration.

In one form of the present invention, the web clients' access to the potential positionee/positionor system can be maintained by sending and receiving requests and results to a web server. All interactive screens are displayed by formatting an HTML page and delivering its content to the user's browser. The web server sends requests for dynamic content to a separate application server. This server accesses the database server to retrieve data, assembles an HTML response, and then delivers the page back to the web server. Batch interface programs execute on the database server to transfer data between the potential positionee/positionor system database and existing mainframe applications. The potential positionee/positionor system application of the present invention, can be broken down into five basic components; the Web site components; the Online application components; the Batch components; the Reporting components; and the Infrastructure Components.

Web site components - The potential positionee/positionor system of the present invention can be accessed by a web browser. All user interface can be handled through the web server by

sending HTML to the client and responding to the client's HTTP requests. The web server can also hold static content such as image files. In addition to the web server, the application server can generate web content. The application server can merge data from the database with HTML to generate the final HTML stream that gets delivered to the client browser. This operation can be performed by a Java Server Page (JSP). A JSP is a HTML page with special Java programming logic embedded in it.

Online application components - The application logic of the present invention of the potential positionee/positionor system can be implemented using iPlanet Application Server (iAS). iAS Servlets implement the majority of the business logic. A servlet is a Java program that executes on the iAS server in the context of a user session. Every user of the potential positionee/positionor system will enter through a logon process. At the time of logon, the user session will be instantiated. From that point forward, each HTTP request from that user that goes to the application servers will execute in the context set up when that user logged on. The Servlet accepts data from the web page where the data was entered. Data validation and database processing is then performed. The process continues with the next JSP being called to present the next page. Another component of the application is stored procedures in the DB2 database server. On the client side, some application logic and special user interface presentation mechanics are handled by JavaScript.

Batch components - Some of the functions performed by the potential positionee/positionor system of the present invention can run at regular intervals and can be scheduled to run in batch mode. These functions are primarily in the area of interfaces to existing mainframe systems. The programs run on the database server. All batch jobs are Korn shell scripts. Inside the script

is the execution of Korn shell commands, Perl Scripts, and COBOL programs, which often make use of stored procedures in the database.

Reporting components - Standard reports are available from the potential positionee/positionor system of the present invention. These are typically daily, weekly, and monthly reports that can be delivered either electronically or manually depending on the capabilities of the individual office. These reports can run in batch mode on the database server.

Infrastructure Components - Functions that are outside of the business logic category of the present invention, but that form a foundation for the inner workings of the system can be classified as infrastructure components. These functions can be responsible for activities such as implementing the security system, error reporting and recovery, and other basic capabilities in the application that can be shared amongst the other components. The infrastructure components for the potential positionee/positionor system can be implemented through the base object model in Java, and extension modules that enhance the capabilities of iAS and the base operating system.

Web Site Architecture

At the top level of the site navigation map illustrated in Figure 1, there is a home page 1. On that page, the user makes a choice to indicate whether they are a job seeker 2 or an employer contract 3. Staff members log onto the system separately from the home page 1 on the staff Login 4. Login procedures require the entry of a valid username and password, or the user is required to complete the registration process to create a username and password. The registration process for the job seeker is illustrated in Figure 2. In order to complete the registration process, Figure 3 illustrates that the job seeker must input skills at the input point 40. The skills List 44 is determined either through a skill search 41 or a hierarchy list 43. The Job Seeker then chooses

their skills from the skill selection sheet 45. Output is through the output point 46. After registration is completed the user can logon to the system. The employer registration 3.1 is illustrated in Figure 4 and also requires the additional step of verifying the employer's registration information 3.2. Once the registration process is completed, the employer may post job order 3.3.

Once the user type is identified and their username and password is authenticated, a customized menu of function options is made available. Figure 5 illustrates one embodiment of the general organization of web pages to provide customized menu of function options. From the Home Page 51, the user type is selected. From either the Job Seeker Menu 52 or the Employer Menu 53 a function is selected. The function is then preformed through a step or series of steps. The step or steps of the selected function may be presented to the user as a single web page, or a series of web pages. A separate URL is required to access the Staff Menu 54.

As illustrated in Figure 1, the Job Seeker may choose the Job Seeker Tab 2.1 and the Qualified Job List 2.3 will automatically present itself from the Job Seeker Tab 2.1, the Job Seeker can choose to Print Registration 2.2. Qualified Job List 2.3 allows the Job Seeker to View Job Information 2.5. A link to a mapping component 2.6 is also provided. There are several components that interplay with the Job Seeker Search Results 2.4. These components include Update Job Seeker 2.7, List Job Seeker Communications 2.8, List Job Seeker 2.9, Add Job Seeker Services 2.10, List Communications 2.11, and Job Seeker Mass Call-In 2.12.

As further illustrated in Figure 1, the registered Employer, once logged into the system, will view the Job Order List 3.3. The Job Order List 3.3 lists the position openings provided by the employer-user. From the Job Oder List 3.3, several functions may be performed. These

functions include: Job Order Statistics 3.30; Job Order Search 3.31; Update Contact Information 3.32; Job Order Tab 3.33; Recruiting Action List 3.34; Qualified Candidate List 3.35; and View Job Seeker Information 3.36. The Job Order Statistics 3.30 function provides statistical information regarding a position opening (job order). The Update Contact Information 3.32 function allows the employer-user to alter contact information for a job order, thereby assuring that the most accurate contact information is presented to the job seeker. Job Order Tab 3.33 allows the employer-user to create a job order. The Recruiting Action List 3.34 function allows the employer-user to determine the actions taken on a job order. Such actions include the recruiting outcome. The Qualified Candidate List 3.35 provides to the employer those job seekers whose skills are compatible with those provided in the job order. The View Job Seeker 3.36 function provides the Employer-user with the information on job seekers contained with the Qualified Candidate List 3.35.

As further illustrated in Figure 1, once a staff member logs into the system, Staff Menu 4.1 is presented. The Staff Menu 4.1 allows the staff member to access all of the Job Seeker and Employer-user functions, as well as the List Employer-user Requested function 4.11, BFS Mirror Search function 4.12, Job Seeker Search function 4.13, Search Employer Information function 4.14. The List Employer Requests function 4.11 allows the staff member user to see the job orders for a particular employer. The BFS Mirror Search function 4.12 permits the staff member user to search the system. The staff member user may also engage the Job Seeker Search function 4.13 to locate a particular Job Seeker's information contained within the system. The Staff Member user may also utilize the Search Employer Information function 4.14 to obtain data regarding a particular employer on the system. The system also allows a staff member user to

update employer contact and job order information, update employer services, and add employer services. This is accomplished through a series of staff-specific web pages.

Figure 6 is an illustration of the general layout of a typical web page for one embodiment of the present invention. The top banner portion 61 provides the title 62 and the logos 63. The top banner portion may also contain various graphics, depicted as Pictures/Images 64. A horizontal strip of global controls 65 is displayed below the top banner portion 61. When applicable, a horizontal strip of task specific controls 66 appears below the global control strip 65. If the page is a list page, a third horizontal strip of menu items related to operating on list screens 67 is available directly below the task specific controls 66. The main body of the typical web page with the primary content 68 follows below the horizontal strips. The main body 68 is where data elements and input/output are performed. A vertical control strip 69 of controls runs along the left hand portion of the main body 68, providing an additional set of global controls. Links to other job related materials are provided on the vertical strip when appropriate to the user type and function being performed.

The pages in the potential positionee/positionor web site can be broken down into five basic categories: the home/logon page, menu pages, search pages, list pages, and detail pages.

Login Page

The home/login page shown in FIGURE 7 is in its own category due to its processing requirements. The initial home page identifies the user type and requests a username and password. At this point, secure sockets layer (SSL) is used for transmitting this information to the web server. Also at this stage, a number of evaluations are performed on the client browser. Once the browser capabilities and the user have been authenticated, a user-appropriate opening

menu page is displayed depending on the user type.

Menu Pages

Menu pages shown in FIGURES 8, 9, and 10 are displayed as appropriate for the type of user, such as job seeker in FIGURE 8, employer in FIGURE 9, and staff in FIGURE 10. A menu page contains no input controls, only links to other pages in the system. Some conditional processing is performed to show or hide specific menu options based on the user's permissions. An example of such conditional processing is the hiding of staff-specific menu options when the user is a job seeker or employer. These decisions are made when the page is constructed on the application server.

Search Pages

Search pages, such as the one shown in FIGURE 11, accept search criteria, and then execute a database search for data with matching criteria. After the search is completed, a list page is built showing the results if one or more matching records is found. If no matching records are found, the search page is redisplayed with an error message.

List Pages

The list pages, such as the one shown in FIGURE 12, list several rows of information from the database. This is typically a result set from a database search. Each result record is a link that can be used to present the detail page for that data row. Optionally, each line in the list may also contain a checkbox that can be used to select a subset of records. The selected set of records can span multiple list pages.

Initially, the result set is divided up into pages. If the result set requires more than one page of list information, navigation buttons will be available to proceed to the next or previous page as

necessary.

When a user selects a detail record, and then returns to the list view, the user will return to the same list page that contained the detail record most recently viewed.

Other activity, such as an employer altering a job order, in the potential positionee/positionor system may introduce or eliminate records from the user's result set. However, once the list is generated, it remains static until the user requests for the information to be refreshed. When necessary, some processes force a refresh to occur.

Detail Pages

When complete detail on a record of information is requested, a detail page, such as the one depicted in FIGURE 13, is presented. If a user requested the detail from a list page, then options on the detail page will exist to move through the list in detail view and an option to return to list view will be in place. If a subset of records was defined on the list page, then that subset defines the context of what the next/previous navigation, such as when a job seeker selects job skills, will present to the user. For example, when a job seeker selects skills, the next navigation can be directed towards providing the jobs that match the job seeker's skills.

In simpler cases, detail pages are displayed from other non-list pages, or used for data entry purposes.

Online Components

Screen Generation

The mechanics of generating a screen begins with the user's browser request. These requests can be sent to one of the potential positionee/positionor system web servers. If the request is for a static HTML page or other static content such as an image, the web server can handle the

request by itself. In one embodiment of the present invention, the only static HTML page is the login page. The rest of the page requests are references to Servlets. In these cases, the requests are forwarded from the web server to the application server that is best suited to handle that request at that time. The best suited server can be determined through load balancing information that flows between the application servers, and from the application servers to the web servers.

The normal processing of an online screen can involve several steps, including:

Building the screen with input fields and any other controls;

Processing the input values, perform validation and database access; and

Providing a response, such as an error message or the presentation of the next screen in the process.

Programmatically, these functions can be split into separate modules. The building of the screen can be performed by a Java Server Page (JSP) for that screen. When the page is submitted for processing by the user, a Java Servlet can be called. After processing the information, the Servlet chains to the next JSP.

In one embodiment of the present invention, after a user is done entering data onto a web page form, some button click or other control function is performed by the user. At this point, validation of data is performed. The potential positionee/positionor system performs validation and enforcement of business logic at three different levels:

On the input form web page itself;

At the application server; and

At the database.

In one embodiment of the present invention, the potential positionee/positionor system can be broken down into five sections:

Job seeker functions;

Employer functions;

Staff functions;

Administration functions; and

Matching functions.

Job Seeker Functions

In one embodiment of the present invention, a job seeker begins his experience with the potential positionee/positionor system by registering, as shown in FIGURE 2. Only registered job seekers with a username and password can use the system, as illustrated in FIGURE 14. Registration is a simple sequence of forms. These forms, shown in FIGURE 15, require that the job seeker input contact information, confidential information, other information, such as the highest level of education completed and willingness to work for temporary agencies, veteran information, and other confidential information. The only field that prevents a person from entering a profile on the system more than once is the field for social security number entry, since duplicate social security numbers are not allowed. FIGURE 16 illustrates veteran information, including a series of questions as well as a checklist of military operations designed to ascertain the veteran status of a job seeker. There are also forms for transportation and work information. The transportation form, shown in FIGURE 17, is designed to allow job seekers to limit matching jobs to those within a certain distance of a zip code. The work information, also shown

in FIGURE 17, is directed at limiting job matches to those jobs which fit the job seekers desired work schedule and salary requirements. Optional forms for work history, FIGURE 18, and education, FIGURE 19, are to provide potential employers with additional information on the job seekers.

The job seeker can also provide the type(s) of positions sought, as illustrated in FIGURE 20, from a hierarchy as well as the skills the job seeker has pertaining to the positions sought. The skills for a given position are predefined and are in the form of a checklist, as shown in FIGURE 21. The skills are further separated by level of experience, also shown in FIGURE 21. This series of forms is used to guide the job seeker through a series of predefined skills in order to add skills to the user's profile. The user selects skills and assigns predetermined proficiency or experience levels to the selected skills. Extensive searching is available to choose skills related to various job titles. This functionality is also available in the employer section to define the required skills for a job order. After the registration procedure is completed, the user can logon to the system by entering their user name and password into the login screen shown in FIGURE 14. Once a job seeker has filled in his skills profile, the matching function can be performed by selecting "Save, Match Me to Jobs Now," as shown in FIGURE 21. Available job orders are then compared with the user and a list of matching job opportunities that correspond to the job requirements is presented. The profile for the job seeker is also saved in the system. Links to the detailed job information is available for matching job orders. Job seekers may also view a map showing the job location. The job seeker can also select "Save, Don't Match Me to Jobs," as shown in FIGURE 21. This also saves the job seeker's profile, but does not match the profile to the job orders. A job seeker can also choose to be removed from the qualified

candidate list. Employers are not notified of the job seeker's non-interest.

Employer Functions

In another embodiment of the present invention, employers must be registered prior to posting any job orders into the system. FIGURE 22 illustrates the employer login requiring a user name and password. FIGURE 23 shows that to obtain a user name and password, the employer must input company information and contact information. This information is then reviewed by organizational staff to determine the validity of the employer. Once the employer goes through the online registration, job order worksheets can be prepared. The job order worksheets, such as the one illustrated in FIGURE 24, contain fields for inputting job information, salary information, benefits information, additional job information, and job posting status. The job information includes fields for entering the job title, job description and duties, tracking identifier for tracking job orders, number of openings, hours per week, shifts available, type of work (full-time, part-time, etc.), and minimum level of education required.

Once this information is entered into the system, the worksite information for the job order is entered. FIGURE 25 depicts the form for entering the worksite information. The worksite information includes fields for entering the job location address, an additional job location address, city, state, and county for the position. The worksite information also allows employers to state whether the position is accessible by public transportation. The worksite information further allows the employer to give permission to the system to provide the job seekers with a map to the position's location. The job order may also provide, at the employer's election, the employer's contact information, as shown in FIGURE 26. Additionally, FIGURE 26 also illustrates that the employer can elect to have daily notifications of new matching job

seekers, or notifications in another time frame, as well as requiring the system to send resumes of job seekers who have indicated an interest in the job order.

The employer may also provide the type(s) of positions sought to be filled, as well as the skills the job seeker has pertaining to the positions sought, as illustrated in FIGURE 27. The skills for a given position are predefined and are in the form of a checklist, as shown in FIGURE 28. The skills are further separated by level of experience, also shown in FIGURE 28. This series of forms is used to guide the employer through a series of predefined skills in order to add skills to the job order's profile. The user selects skills and assigns predetermined proficiency or experience levels desired for the position to the selected skills. Extensive searching is available to choose skills related to various job titles. However, these job orders cannot be posted to the system until the registration has validated the legitimacy of the employer. The job orders may be categorized by their status as not-posted, posted, closed, or on hold/held. A job order that is not posted is a worksheet. The job order worksheet allows the employer to create complete job orders. The posted job order is a complete job order, available for matching. A closed job order cannot be reopened. A job order that is on hold/held will close after a predetermined period. Prior to closure, a notification will go out to the employer regarding the on hold/held job order.

After a job order worksheet is completed, a trial match can be performed. This function allows the employer to determine how many qualified candidates exist in the potential positionee/positionor database, and may be performed prior to the employer's completed registration. No qualified candidate information is available to the employer, other than the number of qualified candidates. Modifications to the job order worksheet can then be performed prior to the actual posting of the job order. These modifications to the job order worksheet can

alter the number of qualified candidates for a job order. Once posted, a match is performed and a list of qualified candidates is generated in the database. A job order can be modified after a match has been generated by the system. If a job order is modified after a match has been generated by the system, the system will generate a new match. The next time a qualified candidate logs onto the system, that new job will appear in their list of qualified jobs.

Once qualified candidates have been identified through the matching process, the employer can perform actions to view the job seeker information and make referrals. An example of the qualified candidate list is illustrated in FIGURE 29. The employer is then free to take action towards the recruiting of qualified candidates. The employer can see the job seeker's information if the job seeker has not labeled such information confidential and the employer takes a recruiting action. Upon taking a recruiting action, the action remains in the recruiting actions list, even if the job seeker never appears on the qualified candidate list again. The recruiting actions, shown in FIGURE 30, trigger notification of the job seeker of a match in skills between them and a job order. Notifications are queued and processed in batch mode. Possible notification methods are an email, automated phone notification, or a letter. If no email address for the employer is provided, then the employer must be staff assisted.

A record of every action conducted by a user is maintained by the system. The system can maintain records on whether the job seeker or employer first expressed interest. If a job seeker expresses interest first, that information is communicated to the employer. If an employer user expresses interest first, that information is communicated to the job seeker. If a "no interest" response is expressed, that information is not communicated.

Staff Functions

In yet another embodiment of the present invention, the staff menu, illustrated in FIGURE 31, is presented when a staff user logs on to the potential positionee/positionor system. The staff menu contains links to every function available to both the job seeker and the employer. Employers can be registered by staff, or employers can submit their own registration requests. The staff member may label the job order as qualifying for a special program, as shown in FIGURE 32. A staff member is the only user able to classify a job order with a special programs designation. Additionally, staff users can identify, and the system will maintain records for, additional service activities provided to the job seeker.

Staff search screens, such as the one illustrated in FIGURE 33, allow the staff member to look up job orders by any one or more of a number of fields. These fields include job order ID, county code, and worksite zip code. Additionally, there is also the ability to search for job seekers, shown in FIGURE 34, through a variety of profile fields. The staff user can also choose to send notification to the job seeker. The job order search screen provides staff members with a method to search for and edit a specific job order. Job order information can also be printed. The staff member may also edit employer company information, edit employer contact (FIGURE 35) and other information, such as transportation information and work information (FIGURE 36) as needed. Employer information can also be printed. Searchable fields denoted with a "+" sign indicate that searches can use multiple search terms. All search results can be printed.

If a job seeker or employer user forgets their password, a staff user can provide a temporary password. Upon entry of the temporary password, the system requires the job seeker or employer user to change the temporary password to a new, permanent password.

Administration Functions

Administrative screens are used to maintain the various basic data of the system such as skills definitions, staff users, security settings, and other table maintenance.

Matching Functions

The present invention is directed towards aligning a job seeker with an employer, in order to facilitate employment. To accomplish this goal, a matching function is required to generate matches between job seekers and employers. The matching application generates matches between job seekers and employers on the basis of job requirements provided by the employer. The job seeker's profile must be identical to the job requirements provided by the employer in order to generate a match. The requirements include the fields of: the highest level of education completed; the willingness to work for temporary agencies; the willingness to travel a distance from a zip code; the kind of work sought; the type of work sought; the shifts available to work; and salary. The skills that were entered independently by both the job seeker as skills held, as shown in FIGURE 21, and the employer as skills required, as shown in FIGURE 28, are used for the purposes of rank-ordering.

The matching application then correlates the job requirements held by the job seekers with those required by the employer to generate job seeker/employer matches. Matches are provided to the employer in the form of a qualified candidate list, as shown in FIGURE 29. Once the employer makes a recruiting action, the job seekers are notified of the matches. The recruiting actions, shown in FIGURE 30, trigger notification of the job seeker of a match in skills between them and a job order. Notifications are queued and processed in batch mode, described below. Possible notification methods are by email, automated phone notification, or a letter.

The matches are also rank-ordered by the relational application. The rank-ordering of

completion of another batch job.

In another preferred embodiment of the present invention, the batch programs for the potential positionee/positionor system run in a Unix environment. In Unix, batch jobs can be either a single executable program, or a shell script. A shell is simply a term for the command line interface to the operating system. Several different shell programs are available in the Unix environment. Examples of these are Bourne, Korn, C, and Bash. Potential positionee/positionor system Batch jobs are written as Korn shell scripts. Korn shell is the most common and popular Unix shell. Within the Korn shell script, individual programs can be executed, environment variables can be used, and basic control structure constructs are available. Return codes from programs can be checked within the script. Return codes from the script can be checked by COSbatch.

In yet another preferred embodiment of the present invention, the core processing of the batch programs is written in Microfocus COBOL.

Two important design features of potential positionee/positionor system batch jobs is their ability to be restartable and their use of checkpoints. Many of the programs in the potential positionee/positionor system will be dealing with large amounts of data. If for some reason the job is interrupted, the ability to restart the job and have it resume processing where it left off saves valuable processing time and reduces performance impact on the system. From an operational standpoint, this approach offers simplicity. Any program can be terminated and restarted without the need for a lengthy manual rollback process.

Central to the checkpoint/restart infrastructure is a batch control table that contains key information about the execution parameters and status of the job. Information contained here

includes input/output file name(s), the current status of the job, and an indicator of where in the file the last checkpoint occurred. There is also a checkpoint governor stored in the table that indicates the number of records to be processed in between checkpoints. This allows for some tuning of resource utilization. This technique limits the number of database locks and the length of time that records stay locked. The checkpoint value is read at the end of each checkpoint interval so that the parameter can be set dynamically.

Many batch programs in the potential positionee/positionor system either generate a data file for the mainframe from data contained in the potential positionee/positionor system, or read a file created on the mainframe and post the information into the potential positionee/positionor system database.

The mechanics of sending and receiving files between the potential positionee/positionor system batch server system and the IBM mainframe consists of dropping files off and reading them from a specific location on the network. In a preferred embodiment, the transfer mechanism is FTP or an NFS mounted volume that can be accessed directly. This is to avoid manual intervention in all file transfers for the potential positionee/positionor system. Files should be dropped off and picked up by the programs automatically with no human intervention.

Infrastructure Components

The potential positionee/positionor system infrastructure can be defined as those components that provide core services to the rest of the application components. In one preferred embodiment of the present invention, the infrastructure centers around iPlanet Application Server. iAS based applications consist of off-the-shelf iAS servers to provide the core services and custom built application components to implement the application's specific business logic

requirements. The custom built application logic components that execute on the server side consist of Java Servlets, Java Server Pages (Java embedded in an HTML document), and application server extensions written in Java and C++.

In another preferred embodiment of the present invention, requests are received from the web user and, via the web server, a specific Servlet is called upon to handle that request. The Servlet can access external resources such as databases. After processing is completed, a Servlet will typically either respond with an HTML stream back to the client, dispatch control to a Java Server Page (JSP), or a combination of the two.

Structuring the iAS application architecture to use separate components for static pages, dynamic page templates, query files, and executable logic provides a multi-tier application model. A great deal of flexibility is available in matching the best module type to the application module's task. The advantages of this scheme are that the application components are separated into manageable pieces according to the skills required to prepare them and by the functions that they perform. This also allows for greater re-use of components, simpler testing, and modular deployment. This supports a higher quality development result and minimizes the impact on system availability when deploying potential positionee/positionor system application software upgrades.

According to another specific embodiment of the invention, the following specific architecture details are part of the potential positionee/positionor system:

Logical Architecture

In one embodiment of the invention, the invention's technical architecture is implemented using a multi-tier architecture illustrated in Figure 38. Users access the potential

positionee/positionor system using standard web browsers which communicate with a collection of web servers. These web servers will serve static content such as static pages and images. Requests for dynamically generated pages will be forwarded to application servers. The dynamic content will be generated via programs invoked by the application servers. These programs will access and manipulate persistent data via a DB2 database on a separate server.

Figure 39 illustrates the integration of components in the potential positionee/positionor system.

Figure 40 shows transaction flow in the potential positionee/positionor system.

In step 1 of Figure 40, the web client requests a resource as specified by a URL. The request is addressed to the IP address obtained from a DNS lookup for a specific host name for the potential positionee/positionor system web presence. The IP address is a virtual IP address associated with the HTTP Router.

In step 2 of Figure 40, the request reaches the Firewall. The Firewall is configured to pass only packets addressed to the HTTP Router's virtual IP address and only to ports 80 (HTTP) and 443 (HTTPS). All other traffic is blocked by the Firewall. The Firewall passes suitable packets to the HTTP Router.

In step 3 of Figure 40, the HTTP Router receives the requests passed on by the Firewall. The HTTP Router selects a web server based on current web server loads and which web servers are suitable for responding to the specific URL requested. The HTTP Router passes the HTTP request on to the selected web server.

In step 4 of Figure 40, the Web Server receives the request passed on by the HTTP Router. The Web Server locates the requested resource. If the resource is static content, then the

Web Server retrieves the contents of the resource and sends it back to the client as an HTTP response. If the request is for dynamic content then the web server forwards the request to an Application Server. The Web Server selects an Application Server based on current server loads and on which Application Servers are suitable for responding to the specific content requested. The Web Server passes the request on to the selected Application Server.

In step 5 of Figure 40, the Application Server receives the request passed on by the Web Server. The Application Server determines which logic module is being requested and invokes it. The logic module may need to access and/or manipulate the database. The module will perform data requests against the Database Server. Data retrieved from the Database Server is used to construct the response.

In step 6 of Figure 40, the Database Server will receive the database transactions from the Application Server. The transactions will be used to invoke and execute queries and stored procedures.

HTTP Router

Ideally, the potential positionee/positionor system should use multiple web servers to accommodate the volume of activity. The challenge of using multiple web servers is in addressing them and in achieving some degree of load balancing. To solve these problems, a device to route HTTP requests among the available web servers can be used. Figure 41 illustrates how this HTTP Router interacts with other components within the potential positionee/positionor technical environment.

The HTTP Router monitors the available web servers to which it is allowed to route traffic in order to determine the availability of the servers – such as if they are up or down and

the amount of load currently on the servers. As new HTTP requests are received from the Internet, the HTTP Router determines which web server to route the message to based on criteria including which servers are available, which servers are least heavily used, and which servers are capable of handling the request for the specific resource. Not all web servers may have all the content. The system may be designed to allow only certain content to be served by only specific web servers.

Each web server has a different IP address. This creates a problem in terms of the URL that the user uses to request resources. Using this HTTP routing scheme, the HTTP Router has a virtual IP address. All requests from the Internet are addressed to that IP address. The appearance from the outside is that the server at that virtual IP address is handling all of the requests.

The use of multiple web servers with an HTTP Router acting as the "front door" makes the architecture very scalable. Additional web servers can be added at a later time and the configuration of the HTTP Router can be updated to include those new web servers.

The use of this scheme has the built in advantage of providing fail-over capabilities. Should a web server go down, the HTTP Router will detect it and not route further traffic to that server. Any transactions being processed by that specific web server at the time of the failure would themselves fail. Due to the structure of the database transactions, data consistency would not be jeopardized. From the user's perspective, the URL request would time-out. If the user would re-request the resource, by clicking the button or link again, then the request would be resubmitted to the HTTP Router and it would direct it to one of the available servers.

Web Servers

The entire potential positionee/positionor application is run from a web browser. All web page content is delivered to the web browsers by the two web servers. Static (non-changing) content is hosted and served directly by the web servers. In the case of requests for content to be generated dynamically, the web servers pass the request through to the application servers and pass the response back through to the web browsers. Figure 42 depicts this flow.

At step 1 of Figure 42, an HTTP request is forwarded on from the HTTP Router to a web server. Each web server is identically configured and has identical capabilities.

At step 2 of Figure 42, the web server receives the request. If the request is for static content then the web server retrieves the content from the file system and returns it through the HTTP Router.

At step 3 of Figure 42, if the request is for dynamic content, then the web server forwards the request to the application server plug-in running within the web server. The plug-in and web server interact via the web server's Application Program Interface (API). The plug-in evaluates the request and selects the optimal application server to send the request to. The plug-in then forwards the request to an application server and receives the response. The response is sent back through the web server.

At step 4 of Figure 42, the plug-in forwards the request to an application server. The application server handles the request, formulates the response, and returns the response to the plug-in.

Ideally, two or more web servers are deployed for the purposes of reliability and performance. Implementing a multi-server solution from the start puts into place the proper infrastructure to scale by adding additional web servers in the future with very little effort.

On each physical web server host computer there will be two web server processes running. One process will service requests for the HTTP protocol providing non-encrypted communications. The other process will service requests for the HTTPS protocol, HTTP running over SSL, providing encrypted communications.

Application Servers

Although the user interface of the potential positionee/positionor system is through a web browser, the system is a robust business application. The application logic is implemented on the application servers. Any database access requests are initiated by the application servers. Figure 43 illustrates the interaction between the application servers and other components.

At step 1 of Figure 43, a request is forwarded on from the HTTP server to an application server. Each web server is capable of determining the optimal application server to send each request to. If an application server becomes non-responsive then the web servers will discontinue forwarding requests to that application server and will send them to the surviving application server instead. Once the application server finishes processing the request, it returns the response back to the web server which sent the request.

At step 2 of Figure 43, the application server receives the request from the web server and begins processing it. The application server will confirm that it is the optimal server to handle that particular request. If that is confirmed, then the application server proceeds with loading and executing the appropriate application logic and constructing the response. The response is sent back to the web server.

At step 3 of Figure 43, in the process of handling the request, the application server may employ the services of the database server to retrieve or update persistent data.

At step 4 of Figure 43, if it determines that another application server is better suited to handle a particular request at that time, then it may forward that request to another application server for processing. Application servers also communicate with each other for purposes of exchanging performance and load balancing information as well as replication user session information.

The application servers are in constant communication with each other to maintain the status of every client's activity. In the event that one application server fails, all of the user sessions with the potential positionee/positionor system would be maintained and carried forward by the remaining server(s).

Like the web servers, implementing two application servers initially provides all of the infrastructure needed to scale performance to higher levels by simply adding an additional application server. Isolating the function of the application server further enhances the ability to improve performance exactly where needed in the future.

Database Servers

The database servers provide the functionality for the data access layer of the application. Batch processing application components are co-hosted with the database services on the same physical servers. The application servers and the batch processing modules are the "clients" to the database services. The batch processing modules exchange information with legacy mainframe applications via periodic interface exports and imports. Figure 44 illustrates this interaction between the database servers and the other system components.

At step 1 of Figure 44, the application servers issue requests for data manipulation to the database servers and receive the data and status back. Both application servers are capable of

communicating with either database server.

At step 2 of Figure 46, the database servers handle the data manipulation requests from the application servers and from the batch processing modules running on the same host computers as the database servers are running on.

At step 3 of Figure 44, the database servers are running within a DB/2 cluster and communicate with each other in order to process queries in parallel and to provide fail-over and a degree of load-balancing. The database servers are also running within a Solaris cluster. The Solaris systems communicate with each other for purposes of fail-over fault-tolerance.

At step 4 of Figure 44, the batch processing modules exchange information with legacy mainframe applications via periodic interface exports and imports.

Ideally, this system should exhibit 99.9% uptime. Therefore, a highly available parallel database server in the system architecture is desirable. For a database to be highly available, single points of failure must be eliminated. This requires that there be at least two systems combined into a database "cluster". In addition to having two physical servers, there must be fault tolerance built into the disk subsystem. The database cluster should be able to survive a disk failure. Connectivity between the cluster members must be redundant to eliminate the cluster communications path as a single point of failure. Figure 45 highlights the major components involved in a highly available database design. Figure 45 also shows four major areas at which failover can occur to achieve high availability. Each of these potential failure points is described below.

A. Database Software/Cluster Software

When two systems are operating together to produce a highly available database, their

software must be equipped to handle faults that may occur. Item A in Figure 45 represents the software components of the cluster. At the operating system and database software level, the systems are aware of each other and coordinate with each other when necessary. The systems also check on each other's health so that they can react when a problem is detected. If one system determines that the other is unable to continue on for some reason, the operating system and database software coordinate to take on the other's workload.

B. Cluster Interconnect Hardware

Item B in Figure 45 represents the hardware used to communicate between the cluster members. Cluster systems use a private communication channel. This keeps the excess traffic off of the general network and often makes use of specialized high speed devices for performance purposes. To keep from having a single point of failure, the cluster is designed with redundant private communication channels so that if one device should fail, the other channel can allow communication to continue.

C. Disk Subsystem

Disk subsystems are at the core of any database system. The loss of a disk drive or even a complete disk cabinet should not cause the system to fail. In item C in Figure 45, both cluster members must have a physical connection to all of the physical disk drives that make up the database so that failover can occur between the systems. The underlying disks also must employ some level of redundancy so that an individual disk drive, controller, or cabinet cannot cause a complete disk subsystem failure. Mirroring or some other level of raid technology is typically employed to achieve this.

D. Network Connection

The database systems must return results back to client systems. Item D in Figure 45 illustrates the use of dual network interface cards (NICs). If one NIC fails, the system can continue to communicate with its clients through the second NIC.

There are several ways of achieving parallelism in a database. The design for the potential positionee/positionor system benefits from the use of at least two physical servers for high availability. A parallel database is defined as a group of two or more database servers dividing the work of presenting a single logical database to the client. This concept is illustrated in Figure 46.

In Figure 46, each server has an active connection to only a subset of the data. The passive connection is available for failover, but is not actively used under normal operating conditions. This is an illustration of a "shared nothing" parallel design. Each system in the database cluster is responsible for a subset of the database.

For a "shared nothing" database to be effective, the data is typically split up between the servers such that half of the users' data is on the disk owned by one system and the other half is connected to the other server. This strategy nets close to twice the performance as a single system. Data access that must access data from both systems is often faster as well. Both systems can collect their portion of the data simultaneously. The system that received the request coordinates collecting the results together to achieve the final result for the client.

The "shared nothing" database design is the most scalable in terms of performance provided the data can be segmented by the user. In one embodiment of the invention, the potential positionee/positionor system design follows this approach.

Physical Architecture

In one embodiment of the invention, the invention's network environment consists of three "subnetworks" as illustrated in Figure 47. These three subnetworks are: the public Internet, a LAN/WAN environment, and the potential positionee/positionor system.

The Internet zone is defined as the portion of the network that includes a link to the Internet, router, firewall, web and FTP servers. Ideally, this should not include the current connection to the Internet for browsing, etc. from the LAN. Also, the system should ideally have at least 10 Mbs of total bandwidth between it and the Internet. Redundancy in this Internet connection should be implemented at the physical link level and at the firewall. The LAN/WAN zone is defined as a LAN environment as well as a WAN connection to remote offices and partner offices. Ideally, the system should have at least 8 Mbs of total bandwidth to and from remote offices on the WAN. The system zone supports communication between servers for the potential positionee/positionor system. This includes the communication that will occur between the web, application, and database servers. The system zone can be subdivided into two virtual LANs (VLANs). One VLAN contains any systems that a user would need to send a packet to (public VLAN). The other VLAN contains the back end systems that perform database and application logic functions (private VLAN). These systems are never contacted directly by an end user. Only the web server contacts these systems in the context of the potential positionee/positionor application. There is a connection from the private system VLAN to the router to allow management and administration workstations to connect to the system servers.

There are three distinct server types involved in the potential positionee/positionor application: web servers, application servers, and database servers. Providing two of each of these systems in the configuration is ideal for fault tolerance and scalability.

Web Server Physical Architecture

In one embodiment of the invention, Sun Solaris 2.6 servers running on a Sparc-II platform are used for the web servers. This is a solid, proven platform for delivering Internet content. A Netscape Enterprise Server (NES) can be used for the web server software platform of the potential positionee/positionor system. NES provides the necessary features and performance necessary to meet the service level goals of the system. NES integrates seamlessly with the application server platform.

Figure 48 is a diagram of the web servers and the components they communicate with. There are redundant communication paths from the end users to the dual HTTP routers. From there, the HTTP traffic is load balanced between the two web servers. Redundant network switches ensure a path to at least one of the web servers even in the event of a switch failure.

Application Server Physical Architecture

In one embodiment of the invention, Sun Solaris 2.6 servers running on a Sparc-II platform are used for the application servers. A iPlanet Application Server (iAS) can be used for the application server component of the potential positionee/positionor system.

iPlanet Application Server provides the application logic, transaction management, data access management, load balancing and security services for the potential positionee/positionor system. Ideally, at least two servers will be deployed. The servers coordinate all user session and overall application state data to provide fault tolerance down to the user level. Even if one server went completely down, no users of the system would lose their session with the sytem, or even the state of any current database transactions.

Figure 49 shows the servers' architecture and interconnection to other components.

Database Server Physical Architecture

In one embodiment of the invention, the IBM DB2 Universal Database Extended Enterprise Edition (UDB EEE) is used for the potential positionee/positionor system's data repository. Sun Solaris 2.6 servers running on a Sparc-II platform using Sun Cluster 2.1 can provide DB2 UDB EEE with the level of performance and reliability the system requires.

DB2 UDB EEE, like many other relational databases, is designed for performance and reliability. Central to these design goals is the use of transaction logging. As modifications are made to the database, a record of each modification is first made in a transaction log. The transaction log will be located in a separate physical area from the rest of the database so that in the event of a database failure, the data can be restored up to the minute by using a previous backup and "replaying" the transaction log. At regular intervals, and at a time when it does not adversely effect performance, transactions are actually committed to the database itself. This technique improves performance since transactions need only be written to the sequential log and updated in memory buffers at the time a transaction commits.

DB2 UDB EEE offers several levels of parallelism. It can take advantage of symetric multiprocessing systems, clustered systems, and a combination of the two. The configuration chosen for the potential positionee/positionor system takes advantage of both techniques.

DB2 UDB EEE is designed to allow multiple physical servers to act as a single logical database. This is accomplished by spreading the data across multiple disks on multiple servers, and taking advantage of the clustering capability of the host operating environment for inter-server communication. Within each server, the database software is capable of dealing with multiple CPUs to divide the workload of multiple clients or complex queries internally.

With DB2 UDB EEE, each server in the cluster can function as both a server and a client. Each server can accept a user database request. If the server has access to all of the data needed, it will satisfy the request by itself. If however some of the data resides on another server, it submits part of the work to itself, and other parts to the other servers for processing in parallel. It then assembles the results from its partners, and returns the complete result to the user client as shown in Figure 50. Figure 50 shows three servers as an illustration of scalability. In the case of the potential postionee/positionor system, the "Database User" is actually the iPlanet Application Server.

DB2 UDB EEEs can be fully integrated with Sun Cluster software. Sun Cluster provides the framework that allows DB2 UDB EEE to provide fault tolerance features for the database in the event of a complete system failure. In one embodiment of the invention, the database design for the potential positionee/positionor system uses two database partitions, one running on each server. These partitions are mirrored by Sun Solaris for fault tolerance at the disk drive level.

The DB2 UDB EEE software comes with a cluster aware agent. This agent registers with the Sun cluster software so that it is notified in the event of a failure of one of the cluster's member systems. When this occurs, the agent handles restarting the partition from the failed system on the surviving system.

The two database servers are both physically connected to two drive array cabinets. Under normal operating conditions, each database server performs reads and writes to a separate set of mirrored drives. The mirror sets are split between the two drive cabinets. Within one drive cabinet, one server uses one set of drives and the other uses a different set of drives.

In the event of a disk failure, the Sun Solaris mirroring software will detect the failure and

operation will continue to the one good mirror set member. Replacement of the bad disk and rebuilding the mirror can be performed without any downtime. In the event of disk controller failure on either of the systems, failover to the remaining good member of the mirror set will occur. Replacement of the bad controller will involve taking the system down, but the other system can continue providing access to both database partitions. In the event of an entire system failing, the Sun Cluster software steps in and enables the surviving system to take control of the mirror set from the failed system.

Figure 51 summarizes the Sun Cluster, mirrored disks, and DB2 UDB EEE's interaction with these components.

Finally, the potential positionee/positionor system should ideally provide tools for configuring each system component as well as real-time status and performance monitoring capabilities.

Infrastructure Architecture

The potential positionee/positionor system infrastructure can be defined as those components that provide core services to the rest of the application components. In one embodiment, much of the infrastructure centers around a iPlanet Application Server.

iAS based applications consist of off-the-shelf iAS servers to provide the core services and custom built application components to implement the application's specific business logic requirements. The custom built application logic components that execute on the server side consist of Java Servlets, Java Server Pages (Java embedded in an HTML document), and application server extensions written in Java and C++.

Servlets and Java Server Pages (JSPs) can use the services provided by the iAS

Extensions. The iAS Extensions function much like assembler exit routines on main frame applications. These extensions extend the core capabilities of the base iAS product to provide such functionality as persistent connections to back-end legacy applications, integration with transaction monitors, integration with third party packages, etc.

Figure 52 illustrates at a high-level how the Servlets, JSPs and Extensions work within a iAS server and the points of interaction with the web server, database servers, and other external services.

Structuring the iAS application architecture to use separate components for static pages, dynamic page templates, query files, and executable logic provides a multi-tier application model. A great deal of flexibility is available in matching the best module type to the application module's task. The advantages of this scheme are that the application components are separated into manageable pieces according to the skills required to prepare them and by the functions that they perform. This also allows for greater re-use of components, simpler testing, and modular deployment. This supports a higher quality development result and minimizes the impact on system availability when deploying potential positionee/positionor application software upgrades. Figure 53 illustrates the tiers of a web application and which iAS and other components address which tier.

Figure 54 shows the flow of a iAS based application.

At step 1 of Figure 54, within a web browser, a user is viewing the "Login" page containing a data entry form. The user enters their user name and password and clicks on the "Login" button.

At step 2 of Figure 54, the request, containing the values entered onto the web form, is

sent through the web server to the application server.

At step 3 of Figure 54, the application server receives the request and runs the "Login" Servlet.

At step 4 of Figure 54, the Servlet retrieves the user's user name and password from the incoming parameters and uses the "Login" query to perform a search within the database to verify those credentials and to retrieve information about this user.

At step 5 of Figure 54, once the credentials have been verified, the Servlet generates a new session identifier and creates a new container (HTTP session object) to hold information pertaining to this user such as the user's user name.

At step 6 of Figure 54, the Servlet then dispatches to the Menu JSP to generate a menu page customized for that user.

At step 7 of Figure 54, as the resulting page is created it is sent back to the web browser via the web server. The new session identifier is also sent to the web browser via an HTTP cookie.

At step 8 of Figure 54, the "Menu" page is received and rendered by the browser. The user can then click on any of the options (links and forms) on that page.

At step 9 of Figure 54, when the user clicks on an option a new request is sent through the web server to the application server. The web browser also sends the session identifier via an HTTP cookie.

At step 10 of Figure 54, the application server receives the request and runs the appropriate Servlet.

At step 11 of Figure 54, the Servlet retrieves all of the incoming parameters, including the

session identifier. The Servlet can then use that session identifier to access the existing HTTP session "object" for that user and modify the information contained within it. The Servlet performs any necessary data access and dispatches to the appropriate JSP to prepare the next page for the user.

At step 12 of Figure 54, optionally, the JSP can make necessary calls to the database to retrieve additional data.

Security Architecture

In one embodiment of the invention, a security architecture provides safeguards to protect, detect, and recover from security breaches. Due to the nature of the public environment and infrastructure, web sites and web-based applications are exposed to several security threats, such as communications eavesdropping, communications tampering, host system breach and authorization violations, denial of service, data and software integrity, and second party repudiation of business transactions.

These security issues can be addressed by this architecture in several ways, including access control, authentication, authorization, confidentiality services, data integrity services, non-repudiation services, intrusion detection, attack recovery, and service protection.

The security architecture can further be broken down into six categories: network safeguards, host server safeguards, Web and FTP server safeguards, application server safeguards, database and batch server safeguards, and system application safeguards.

Network Safeguards

The network architecture has checkpoints at which traffic flow can be denied. This effectively divides the network into sub-networks or zones. Figures 55 and 56 illustrate the

delineation made between these zones. The Fire Wall provides network packet level access control to the internal network and the servers. The Filtering Router functions much like a fire wall between sub-nets within the network.

The Fire Wall allows the following: Incoming HTTP (web) and FTP (file transfer) traffic to the Web & FTP Servers, and Outgoing SMTP (e-Mail) traffic from the Web & FTP Servers. All other communications will be blocked at the Fire Wall.

The Filtering Router allows the following: Incoming HTTP (web) traffic to the Web & FTP Servers, Incoming and Outgoing traffic between the Database & Batch Servers and the SNA Gateways for purposes of exchanging files for the various interfaces, and Incoming and Outgoing traffic between the Web & FTP, Application, and Database & Batch Servers and various workstations for purposes of system administration and management. Protocols used will include HTTP, SNMP, FTP, Telnet, Netscape Communication Protocol, and RCP. All other traffic will be blocked.

Packet routing authorization is performed based on source IP address, destination IP address, and protocol (as indicated by the destination port number). These items are enforced by the IP protocol and are fundamental to packet routing and delivery. If an external intruder tampered with either address in an attempt to evade these safeguards, the packet would either become undeliverable or any result would not be deliverable back to the intruder.

In one embodiment of the invention, a fire wall consisting of CheckPoint-1 fire wall software running on a Solaris 2.5 operating system running on a Sun Sparc 5 Workstation will be used. This fire wall is well sized for the potential positionee/positionor system. Control and configuration of the Fire Wall is controlled through user authentication, authorization, and access

control. User authentication is done via user IDs and passwords which are stored in a standalone, self contained user database on the fire wall host computer. Access to control and configure the fire wall is restricted to only those identified and authorized users. The fire wall host computer is not used for any other purpose. Control and configuration of the other network components are controlled through user authentication, authorization, and access control enforced by the individual components. Authentication is done via user ID and password.

The term "hardened" with regard to computer security refers to components which do not use commercially available operating systems and provide limited or no interactive login. Basically, these are "black boxes". In one embodiment of the invention, the potential positionee/positionor system uses hardened network components such as routers and switches for the networking infrastructure. This greatly limits the potential for break-ins and data or configuration corruption for these components.

Use of redundant components provides for higher availability through fail-over in the event of a component failure. The failure might occur through a malicious assault or by "natural causes."

The networking infrastructure is monitored via the SNMP protocol through automated tools. These same tools allow the potential positionee/positionor system Administrator to control and manage the network components.

Host Server Safeguards

The "servers" consist of (at least) two components: the software application providing the service functionality (software service) and the host computer on which this software runs (host server). Security safeguards are implemented on and by the host servers. These safeguards

necessarily provide protection to the applications running on them.

Access to the computer servers on which the software services run is restricted. Methods of access to the host computers include Telnet, FTP, rcp, rlogin, SNMP, and Netscape Communication Protocol (NCP). Authentication is made by user ID and password which are verified against a user database local to each individual host system.

Access control restricts access to system resources such as entries within the file system, use of commands and software, network ports, and other resources. Limiting access to the underlying files and file system that hold the programs and data that support the software services, including the Web, Application, and Database Servers, provides enhanced confidentiality and integrity for those components. This also provides protection for the operating system software and configuration. Interactive logon to the host systems is for support and administration purposes only and is greatly restricted. User passwords are set to expire periodically.

Authentication, authorization, access control, and password policies are enforced by the UNIX operating system. UNIX provides a high degree of security and a fine granularity of control over access permissions assigned by user ID, group membership, resource being accessed, and the type of access allowed. In addition, UNIX is a highly stable and robust operating system with wide support. This provides a stable and reliable environment for the potential positionee/positionor system thus increasing availability.

The operating system, application software, custom potential positionee/positionor application, and supporting configuration and data files are backed up on a daily basis. This provides recover-ability in the event of the loss or corruption of this information through either

an assault or as a result of component failure. This helps to ensure availability and integrity.

System activity and access is logged by the operating system and associated utilities. Log file access is restricted to Owner=Read+Write, Group=Read, World=No Access. Log files are rotated daily. Security tools are used to analyze the previous day's logs. The previous month's logs are archived.

The host systems are monitored via the SNMP protocol through automated tools. These same tools allow the potential positionee/positionor system Administrator to control and manage the host systems.

Automated tools monitor key files, including the operating system, configurations, and applications in order to detect unexpected changes. This provides a means to detect intrusions and protect the integrity of the application.

Physical security facilities and policies to protect against fire, smoke, explosions, humidity, dust, earthquake, storms, other natural disasters, vibration, food and drink, and theft and vandalism are beneficial. Adequate ventilation and cooling should also be provided.

The host system configurations employ redundant and hot-swapable components. This helps to ensure availability of services. These measures include: disk mirroring, hot swapable disk, N+1 redundancy of power and cooling modules, hot swapable power and cooling modules, and selective N+1 redundancy of network interfaces.

Web and FTP Server Safeguards

Access to control and configure the Web & FTP Servers, as well as to retrieve the resources served by them, are restricted.

The following HTTP request methods will be allowed from the Internet:

GET: requests a document or other resource from a specific location

HEAD: functionally like GET, except that the server will reply with only the header information about the resource (such as size, name, author, last date modified, etc.) but won't return the actual content.

POST: allows the client to specify data to be sent to some data-handling program that the server can access. This data is sent in the request body.

Other HTTP request methods such as PUT and DELETE will not be supported. Access control also restricts which resources (URLs) can be requested by a specific request.

Individual users are tracked by the potential positionee/positionor application and not by the Web Server. However, the Web Server restricts access to resources based on identification of the requesting client computer. This is done to identify users attempting to access the potential positionee/positionor system from "privileged" workstations. Staff functions are supported by the potential positionee/positionor system only if the user is authenticated as "STAFF" by the potential positionee/positionor application and if they access the system from a "privileged" workstation. These workstations are identified by the Web Server as having either an internal IP address or by presenting a valid and trusted X.509 digital certificate. Anonymous FTP access is disabled.

Access to control and configure the Web and FTP Servers is controlled through mechanisms separate from the authentication and access control for web content.

Requests and responses carrying sensitive or critical data are sent using HTTP over SSL (HTTPS) using encryption and integrity checking. This provides confidentiality, ensures the integrity of the communication, and provides the user with independent certification of the web

site identity. This protects against assault schemes such as "man-in-the-middle" and "transaction replay". Web Browsers supporting SSL are readily available via free download from the Web Browser vendors, or at low cost through retail channels. Web Browsers that do not support SSL are not allowed to access or transmit sensitive content.

The Web Servers are monitored via the SNMP protocol through automated tools. These same tools allow the potential positionee/positionor system Administrator to control and manage the Web Servers.

Web Server access and errors are logged by the Web Servers. Log file access is restricted to Owner=Read+Write, Group=Read, World=No Access. Log files are rotated daily. Security tools are used to analyze the previous day's logs. The previous month's logs are archived.

The Web service processes run under non-privileged user accounts on the host server. They have restricted access to the file system thus restricting which files they can access and modify. This is particularly important since the Web Servers are used to retrieve files from the host system and can be used to invoke programs on the host system.

The Web and FTP Servers are configured to serve content out of mutually exclusive directories. There is no overlap between the directories used by the Web Servers, FTP Servers, and the underlying operating system. This restricts these services from allowing "back door" access to read or modify key files.

CGI program execution on the Web Servers can be disabled except as required for interaction with the Application Server to limit the ability of an intruder to introduce and execute their own program via the Web Server.

When an HTTP request is made for a resource directory without specifying a specific file,

the Web Server looks within that directory for a document named "index.htm*" or "home.htm*". If such a document is found then it is returned in the HTTP response. If no such document is found then the Web Server can be configured to create a directory listing page. This feature is disabled. This prevents a web user from perusing through directories.

Ideally, the host system on which the Web and FTP Servers run is used exclusively for the purpose of hosting these services. No other application software should run on these computers and no other users or administrators should have direct access to these computers.

Web Server content is maintained only via protocols which are not permitted through the fire wall. These protocols include Telnet, NFS, rcp, and Netscape Communication Protocol (NCP, a proprietary protocol of the Application Server).

Redundant Web and FTP Servers are used to provide load balancing and fail-over. The TCP Traffic Router directs web traffic to the pool of available servers, routing traffic around servers that are non-responsive.

Application Server Safeguards

Access to control and configure the Application Server is restricted. Authentication is made by user IDs and passwords and is controlled by the Application Server administration services. This same mechanism is used to control deployment of application components to the Application Server. The networking infrastructure prevents direct access from the Internet to the Application Server. Access control to the potential positionee/positionor application is controlled by the application.

The Application Servers are monitored via the SNMP protocol and proprietary protocols through automated tools. These same tools allow the potential positionee/positionor system

Administrator to control and manage the Application Servers.

Application Server access and errors as well as application messages are logged by the Application Server. Log file access is restricted to Owner=Read+Write, Group=Read, World=No Access. Log files are rotated daily. Security tools are used to analyze the previous day's logs. The previous month's logs are archived.

The Application Server processes run under non-privileged user accounts on the host server. They have restricted access to the file system thus restricting which files they can access and modify.

Ideally, the host system on which the Application Server runs is used exclusively for the purpose of hosting these services. No other application software should run on these computers and no other users or administrators should have direct access to these computers.

Application Server content is maintained only via protocols which are not permitted through the fire wall. These protocols include FTP, Telnet, rcp, and Netscape Communication Protocol (NCP, a proprietary protocol of the Application Server).

Redundant Application Servers are used to provide load balancing and fail-over. The Application Server software provides this feature.

Database and Batch Server Safeguards

Access to control and configure the Database Server is restricted. Authentication is made by user IDs and passwords and is controlled by the Database Server administration services. This same mechanism is used to control deployment of stored procedures and data definition changes to the Database Server. The networking infrastructure prevents direct access from the Internet to the Application Server.

Access to query and manipulate the data is also restricted. Authentication is made by user IDs and passwords and is controlled by the Database Server administration services. Access to modify data is restricted to being done only through stored procedures. Stored procedures are limited to the rights of the user ID associated with the database connection on which the stored procedure is run. Performing all updates through stored procedures helps to ensure the integrity of data manipulation operations.

The user credentials necessary to establish a connection to the database, and to query and manipulate the data, are completely separate from the user credentials presented by the end users when challenged by the potential positionee/positionor application. This separation of user domains helps to obscure the user IDs that can be used to access the database. Database user IDs are used by the potential positionee/positionor application software on behalf of end users but are not known to, or used by, the end users directly.

The Database Servers are monitored via the SNMP protocol and proprietary protocols through automated tools. These same tools allow potential positionee/positionor system Administrators to control and manage the Database Servers.

Ideally, the host system on which the database and batch Server runs is used exclusively for the purpose of hosting these services. No other application software should run on these computers and no other users or administrators should have direct access to these computers.

Database Server content is maintained only via protocols which are not permitted through the fire wall. These protocols include FTP, Telnet, rcp, and proprietary protocols of the Database Server.

Redundant Database Servers are used to provide load balancing and fail-over. The

Database Server software along with the underlying operating system provides this feature.

The identifying credential of users performing data updates is recorded as audit information within the updated database records. Additional audit information such as the date and time of the update are also recorded. Assuming that the user's credentials and the potential positionee/positionor application have not been compromised, the user will not be able to repudiate the transaction.

System Application Safeguards

Access to use the potential positionee/positionor System is restricted. Authentication is made by user IDs and passwords and is controlled by the potential positionee/positionor application software. User IDs and passwords are stored in the database and used to authenticate credentials presented via web forms by end users.

The System recognizes distinct user types including: Employer, Job Seeker, Staff, Application Administrator, and System Administrator. Employers are able to view all of their own data, update most of their own data, and view the public information of Job Seekers. Job Seekers are able to view all of their own data, update most of their own data, and view the public information of Employers. Staff are able to view all information and update most information of all Employers and Job Seekers. Some restrictions apply to support the concept of Account Managers. Application Administrators have the same privileges as Staff but are also able to change the base data of the System including Clusters, Groups, Titles, and Skills as well as various code tables. System Administrators are able to monitor, control, and manage the System but do not have any specific access to view or change application data including Employers, Job Seekers, Job Orders, base data, etc. However, due to the fact that the System Administrators will

have access to the data and program files in order to perform maintenance and backups, they will have the resources to directly access those files.

The type of user privileges allowed are determined by the user ID presented by the end user. Staff functions are also restricted in that the end user must be accessing the system from a privileged workstation.

The potential positionee/positionor application is monitored via the SNMP protocol and proprietary protocols through automated tools. These same tools allow the System Administrator to control and manage the application.

Potential positionee/positionor application access and errors are logged by the application. Log file access is restricted to Owner=Read+Write, Group=Read, World=No Access. Log files are rotated daily. Security tools are used to analyze the previous day's logs. The previous month's logs are archived.

Ideally, the host system on which the database and batch Server runs is used exclusively for this purpose. No other application software should run on these computers and no other users or administrators should have access to these computers.

The potential positionee/positionor application is designed to take advantage of the fail-over and load balancing capabilities of the underlying Application Server, Database Server, and host systems.

Data entry values are validated within the HTML form within the Web Browser and again by the application running within the Application Server. The client-side validation is a convenience to provide quick feedback for common data entry errors and to avoid unnecessary network and system resource consumption. However, HTTP requests can be spoofed, therefore,

any input values received via HTTP requests go through server-side validation.

When a user clicks on a web link or otherwise causes the Web Browser to issue an HTTP request, the URL of the previous page is sent in the HTTP request header as the "HTTP Referrer" variable. The potential positionee/positionor application checks this value on each page request to ensure that the user is navigating the system in the proper sequence and has not used other navigation means (such as the Web Browser "back" and "forward" buttons, Web Browser bookmarks, or direct URL entry) to go to pages out of sequence.

There are two HTTP methods for making an HTTP request and including data: POST and GET. The GET method sends the data as a query string appended to the URL which is sent in the request header. The POST method accommodates sending the data within the body of the request. HTML hyper-links can use only the GET method. HTML forms can use either the GET or POST method. With the GET method, the data, possibly including sensitive information, is included on the URL. This presents security concerns such as the visibility of the URL for the current page on the Web Broswer user interface.

The potential positionee/positionor application uses only the POST method for HTML forms submission. For HTML hyper-links, no sensitive information is included in the query string. In some cases, this means using an HTML form with only a single button when an HTML hyper-link would otherwise have been used.

Since the source code of HTML pages and JavaScript sent to a Web Browser can be viewed by the end user, no internal application values are sent in these pages. These values include database record keys, database table and field names, host names and internal IP addresses, and internal user IDs such as for database access.

According to one specific embodiment of the invention, Figures 57, 58, 59, and 60 represent the potential positionee/positionor system's database. The following tables describe the components of the database:

Table List

Name	Code	Number
ACTIVE USER	BSSM095T	200
BATCH FREQUENCY	BSSM086T	10
BFS EMP	BSSM001T	350000
BFS FORMS	BSSM070T	10
CNTY	BSSM002T	100
CODE LOOKUP DATA	BSSM090T	2000
CODE LOOKUP HDR	BSSM033T	200
COMMUNICATIONS	BSSM003T	6000000
CONV SKILL KEYWORD	BSSM990T	18000
CORP EMP	BSSM004T	60000
DATE TIME MATH	BSSM005T	1
DESC	BSSM006T	16000
DOT CODE	BSSM092T	1000
DUP SERVLET TMST	BSSM094T	1000
EMP CONTACT	BSSM012T	70000
EMP SRVCS	BSSM010T	100
EMP SRVCS PRVDED	BSSM011T	500000
ERR CODES	BSSM013T	500
HIER	BSSM015T	1000
HIER SKILL	BSSM016T	20000
IETC OFFICE	BSSM018T	100
IETC PARTNERS	BSSM01 9T	100
JO	BSSM020T	240000
JO BENEFITS	BSSM088T	1200000
JO SKILL	BSSM022T	4000000
JO SPC PGMS	BSSM017T	120000
JO STAT	BSSM021T	720000
JS	BSSM023T	3000000
JS CASE MGR	BSSM089T	300000
JS EDUCATION	BSSM067T	9000000
JS RIBBON	BSSM008T	3000
JS SKILL	BSSM024T	60500000
JS SPC PGMS	BSSM025T	60000
JS SRVC PROVIDED	BSSM026T	12000000
JS SRVCS	BSSM027T	100
JS STAT	BSSM028T	9000000
JS WRK HSTRY	BSSM029T	9000000
MAIL FORM	BSSM035T	20
MAIL FORM LO	BSSM036T	2000
NEW HIRE	BSSM014T	5000000
OCCUP SKILL	BSSM037T	15000
OFFICE_ZIPS	BSSM039T	1000
PARTNER OFFICE	BSSM040T	100
PHONE MESSAGE	BSSM041T	500
PHONE MESSAGE LO	BSSM042T	500
PROCESS CONTROL	BSSM076T	4000

PROCESS FILE	BSSM077T	6000
PROGRAM GROUP	BSSM030T	250
PROGRAM GROUP NAV	BSSM031T	1000
PROGRAM YEAR	BSSM068T	1
QUAL CANDIDATE	BSSM043T	7300000
REASON CODE	BSSM044T	200
REGION	BSSM096T	10
RFRL ACTION	BSSM046T	3000000
RIBBON	BSSM007T	200
RIGHTS	BSSM072T	5
SDA ZIP CODE	BSSM093T	100
SEQ NUM	BSSM048T	100
SESSION SKILL	BSSM032T	10000
SIC CODE	BSSM049T	40000
SKILL QUANT	BSSM050T	20
SKILL QUANT TYPE	BSSM051T	20
SOC CODE	BSSM091T	1000
SPCL PGMS	BSSM053T	20
SRVC DLVRY AREA	BSSM054T	30
ST	BSSM055T	50
STAFF	BSSM056T	1500
TRAVEL DIST	BSSM009T	10
UI CLMNT HSTRY	BSSM058T	00000
USER HISTORY	BSSM057T	3075000
USER LOGIN	BSSM059T	3075000
USER RIGHTS	BSSM034T	3075000
USER TYPES	BSSM060T	10
USER_COMM_MODE	BSSM066T	30
VET STAT	BSSM061T	10
WELFARE HSTRY	BSSM062T	400000
ZIP CODE	BSSM063T	7000
ZIP PROXIMITY	IBSSM065T	2550000

BSSM001T
Name: BFS EMP
Code: BSSM001T
Label:
Owner:
Number: 350000
PIK constraint:
Source: Entity BFS EMPLOYER

Options
in
BSSMOOIS
{
index in
BSSM0010
}

Description
This is a mirror of the BFS Employer Table. Only the relevant columns to the potential positionee/positionor system are extracted and put on this table. It will be used to help fill in information for contacts.

Column List

Name	Code	Type	P	M
ID UI ACCT NUM	ID_UI_ACCT_NUM	INTEGER	Yes	Yes
ID FEIN	ID_FEIN	INTEGER	No	Yes
TEXT EMPLR NAME	TEXT_EMPLR_NAME	CHAR(40)	No	Yes
TEXT EMPLR NAME UP	TEXT_EMPLR_NAME_UP	CHAR(40)	No	Yes
TEXT DBA	TEXT_DBA	CHAR(40)	No	Yes
TEXT DBA UP	TEXT_DBA_UP	CHAR(40)	No	Yes
TEXT ADDR1	TEXT_ADDR1	CHAR(35)	No	Yes
TEXT ADDR2	TEXT_ADDR2	CHAR(35)	No	Yes
TEXT CITY	TEXT_CITY	CHAR(18)	No	Yes
TEXT CITY UP	TEXT_CITY_UP_CITY_UP	CHAR(18)	No	Yes
CODE ST	CODE_ST	CHAR(2)	No	Yes
CODE ZIP	CODE_ZIP	INTEGER	No	Yes
CODE ZIP PLUS4	CODE_ZIP_PLUS4	INTEGER	No	Yes
TEXT CNTRY	TEXT_CNTRY	CHAR(35)	No	Yes
TEXT CNTY NAME	TEXT_CNTY_NAME	CHAR(35)	No	Yes
CODE OWNR TYPE	CODE_OWNR_TYPE	INTEGER	No	No
CODE SIC	CODE_SIC	SMALLINT	No	Yes
CODE EMPLR STATUS	CODE_EMPLR_STATUS	CHAR(1)	No	Yes

BSSM002T
Name: CNTY
Code: BSSM002T
Label:
Owner:
Number: 100
PIK constraint:
Source: Entity COUNTY

Options
in
BSSMOOCS
{
index in
BSSMOOCO
}

Description
County is a governmental geographic boundary.

Column List

Name	Code	Type	P	M
CODE CNTY	CODE_CNTY	SMALLINT	Yes	Yes
ID OFFICE	ID_OFFICE	SMALLINT	No	Yes
ID SRVC DLVRY AREA	ID_SRVC_DLVRY_AREA	CHAR(2)	No	Yes
CODE ST	CODE_ST	CHAR(2)	No	Yes
TEXT CNTY NAME	TEXT_CNTY_NAME	CHAR(35)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
MST CREATED	MST_CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	MST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM003T

Name: COMMUNICATIONS
Code: BSSM003T
Label:
Owner:
Number: 6000000
PIK constraint:
Source: Entity COMMUNICATIONS

Options
in BSSM003S
{
index in
BSSM0030
}

Description
This table stores any communication that is being or has been sent to the designated user. It will use the tmst_processed to know whether the communication has been processed.

Column List

Name	Code	Type	P	M
ID COMM	ID_COMM	INTEGER	Yes	Yes
ID REASON	ID_REASON	SMALLINT	No	Yes
ID USER	ID_USER	INTEGER	No	Yes
CODE COMM MODE	CODE_COMM_MODE	INTEGER	No	Yes
CODE REQUEST	CODE_REQUEST	SMALLINT	No	Yes
CODE PROC PGM NAME	CODE_PROC_PGM_NAME	INTEGER	No	Yes
MST PROCESSED	MST_PROCESSED	TIMESTAMP	No	No
ID ISM USER FROM	ID_ISM_USER_FROM	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	MST_CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	MST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM004T
Name: CORP EMP
Code: BSSM004T
Label:
Owner:
Number: 60000
PK constraint:
Source: Entity CORP EMPLOYER

Options
in BSSM004S
{
index in
BSSM0040
}

Description
Contains the information necessary to register an employer to the potential positionee/positionor system at the parent company level.

Column List

Name	Code	Type	P	M
ID ISM EMP	ID_ISM_EMP	INTEGER	Yes	Yes
CODE SIC	CODE_SIC	CHAR(5)	No	Yes
ID UI ACCT NUM	ID_UI_ACCT_NUM	INTEGER	No	No
NAME CORP EMP	NAME_CORP_EMP	CHAR(40)	No	Yes
NAME CORP EMP UP	NAME	CHAR(40)	No	Yes
TEXT DBA NAME	TEXT_DBA_NAME	CHAR(40)	No	Yes
TEXT DBA NAME UP	TEXT_DBA_NAME_UP	CHAR(40)	No	Yes
FLAG FED CONTRACT	FLAG_FED_CONTRACT	CHAR(1)	No	Yes
CODE OWNR TYPE	CODE_OWNR_TYPE	INTEGER	No	Yes
ID FEIN	ID_FEIN	INTEGER	No	No
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
MST CREATED	MST_CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	MST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM005T
Name: DATE TIME MATH
Code: BSSM005T
Label:
Owner:
Number: 1
PIK constraint:
Source:

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Description
This table contains one row and provides any easy way to do arithematic on dates and times in a program. Since there is only one row a basic select will make the necessary date computations and not have to worry about returning more than one row.

Column List

Name	Code	Type	P	M
ID DT MATH	ID_DT_MATH	INTEGER	Yes	Yes

BSSM006T
Name: DESC
Code: BSSM006T
Label:
Owner:
Number: 16000
PK constraint:
Source: Entity DESCRIPTION

Options
in BSSM006S
{
index in BSSM0060
}

Description
This is a description of a spot in the hierarchy of skills.

Column List

Name	Code	Type	P	M
ID DESC	ID_DESC	INTEGER	Yes	Yes
ID HIER	ID_HIER	INTEGER	No	Yes
ID SKILL	ID_SKILL	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
MST CREATED	MST_CREATED	IMESTAMP	No	Yes
MST LAST UPDATE	MST_LAST_UPDATE	TIMESTAMP	No	Yes
TEXT DESC NAME	TEXT_DESC_NAME	VARCHAR(255)	No	Yes
TEXT DESC	TEXT_DESC	VARCHAR(2000)	No	Yes
TEXT ALIASES	TEXT _ALIASES	VARCHAR(13750)	No	Yes

BSSM007T

Name: RIBBON

Code: BSSM007T

Label:

Owner:

Number: 200

PK constraint:

Source:

Options

in BSSMOOCS

{

index in BSSMOOCO

}

Description

This table lists all of the ribbons that a veteran can have.

Column List

Name	Code	Type	P	M
ID RIBBON	ID_RIBBON	INTEGER	Yes	Yes
DATE ISSUE START	DATE_ISSUE_START	DATE	No	Yes
DATE ISSUE END	DATE_ISSUE_END	DATE	No	Yes
TEXT RIBBON NAME	TEXT_RIBBON_NAME	CHAR(100)	No	Yes
ID ISM USER	ID _ISM_USER	INTEGER	No	Yes
TMST LAST UPDATE	MST_LAST_UPDATE	IMESTAMP	No	Yes

BSSM008T

Name: JS RIBBON

Code: BSSM008T
Label:
Owner:
Number: 3000
PK constraint:
Source:

Options
in BSSM008S
{
index in BSSM0080
}
partitioning key (ID_USER
)

Description

This table contains all of the Armed Services ribbons that a job seeker has received.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
ID RIBBON	ID_RIBBON	INTEGER	Yes	Yes
MST CREATED	MST_CREATED	TIMESTAMP	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
MST LAST UPDATE	MST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM009T
Name: TRAVEL DIST
Code: BSSM009T
Label: Travel Distance Code Table
Owner:
Number: 10
PIK constraint:
Source:

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Description
This code table contains the distance code and the text descripition with the range of miles. It is separated out of the standard code table because we want to know the maximum miles from the range.

Column List

Name	Code	Type	P	M
CODE_TRAVEL_DIST	CODE_TRAVEL_DIST	INTEGER	Yes	Yes
TEXT TRAVEL 151ST	TEXT_TRAVEL	CHAR(40)	No	Yes
NUM TRAVEL DISTNCE	NUM_TRAVEL_DISTNCE	NUMERIC(3)	No	Yes

BSSM010T

Name: EMP SRVCS
Code: BSSM010T
Label:
Owner:
Number: 100
PIK constraint:
Source: Entity EMPLOYER SERVICES

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Description

These are the services that a staff member can do.

Column List

Name	Code	Type	P	M
ID SRVC	ID_SRVC	INTEGER	Yes	Yes
TEXT SRVC	TEXT_SRVC	CHAR(50)	No	Yes
FLAG ISM SRVCS	FLAG	CHAR(1)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
MST CREATED	MST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	MST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM011T

Name: EMP SRVCS PRVDED
Code: BSSM011T
Label:
Owner:
Number: 500000
PK constraint: BSSM0110
Source:

Options
in BSSMO11S
{
index in BSSM0110
}

Description
This contains all of the services that a staff member has done on the specified day.

Column List

Name	Code	Type	P	M
ID SRVCS PRVDED	ID_SRVCS_PRVDED	INTEGER	Yes	Yes
ID USER	ID_USER	INTEGER	No	Yes
ID SRVC	ID_SRVC	INTEGER	No	Yes
DATE SRVC	DATE	DATE	No	Yes
FLAG DELETE	FLAG_DELETE	CHAR(1)	No	Yes
ID EMP USER	ID_EMP_USER	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
MST CREATED	MST_CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	MST_LAST_UPDATE	TIMESTAMP	No	Yes
TEXT COMMENT	TEXT_COMMENT	VARCHAR(255)	No	Yes

BSSM012T

Name: EMP CONTACT
Code: BSSM012T
Label:
Owner:
Number: 70000
PK constraint:
Source: Entity EMPLOYER CONTACT

Options
in BSSM012S
{
index in BSSM0120
}

Description
Contains the information necessary to register an employer to the potential positionee/positionor at a given location.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
ID ISM EMP	ID_ISM_EMP	INTEGER	No	No
CODE CNTY	CODE_CNTY	SMALLINT	No	Yes
CODE ST	CODE_ST	CHAR(2)	No	Yes
CODE SALUT	CODE_SALUT	INTEGER	No	Yes
NAME FIRST	NAME_FIRST	CHAR(20)	No	Yes
NAME FIRST UP	NAME_FIRST_UP	CHAR(20)	No	Yes
NAME MIDDLE INIT	NAME_MIDDLE_INIT	CHAR(1)	No	Yes
NAME LAST	NAME_LAST	CHAR(40)	No	Yes
NAME LAST UP	NAME_LAST_UP	CHAR(40)	No	Yes
CODE SUFFIX	CODE_SUFFIX	INTEGER	No	Yes
TEXT EMPLR NAME	EXT_EMPLR_NAME	CHAR(40)	No	Yes
TEXT EMPLR NAME UP	EXT_EMPLR_NAME_UP	CHAR(40)	No	Yes
CODE OWNR TYPE	CODE_OWNR_TYPE	INTEGER	No	Yes
ID UI ACCT NUM	ID_UI_ACCT_NUM	INTEGER	No	No
ID FEIN	ID_FEIN	INTEGER	No	No
TEXT ADDR1	TEXT_ADDR1	CHAR(35)	No	Yes
TEXT ADDR2	TEXT_ADDR2	CHAR(35)	No	Yes
TEXT CITY Y	TEXT_CITY	CHAR(1 8)	No	Yes
TEXT CITY UP P	TEXT_CITY_UP	CHAR(18)	No	Yes
CODE ZIP	CODE_ZIP	INTEGER	No	Yes

TEXT PHN NUM	EXT_PHN_NUM	CHAR(10)	No	Yes
TEXT PHN EXT	TEXT_PHN_EXT	CHAR(6)	No	Yes
TEXT FAX PHN NUM	TEXT_FAX_PHN_NUM	CHAR(10)	No	Yes
TEXT EMAIL ID	TEXT_EMAIL_ID	CHAR(40)	No	Yes
TEXT EMAIL ID UP	TEXT_EMAIL_ID_UP	CHAR(40)	No	Yes
TEXT EMP TITLE	TEXT_EMP_TITLE	CHAR(40)	No	Yes
TEXT EMP DEPT	TEXT_EMP_DEPT	CHAR(40)	No	Yes
ID ACCT EXEC	ID_ACCT_EXEC	INTEGER	No	No
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
ID CREATED USER	ID_CREATED_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM013T
Name: ERR CODES
Code: BSSM013T
Label:
Owner:
Number: 500
PIK constraint:
Source:

Options
in BSSM00CS
{
index in BSSM00C00
}

Description
This is a table of all the error messages in the system.

Column List

Name	Code	Type	P	M
CODE ERR	CODE_ERR	INTEGER	Yes	Yes
TEXT ERR	TEXT_ERR	VARCHAR(255)	No	No

BSSM014T
Name: NEW HIRE
Code: BSSM014T
Label:
Owner:
Number: 5000000
PK constraint: BSSM0140
Source:

Options
in BSSM014S
{
index in BSSM0140
}

Column List

Name	Code	Type	P	M
ID SSN	ID_SSN	CHAR(9)	Yes	Yes
ID FEIN	ID_FEIN	INTEGER	Yes	Yes

TMST CREATED	TMST_CREATED	TIMESTAMP	Yes	Yes
DATE OF HIRE	DATE_OF_HIRE	DATE	No	No
CODE ZIP	CODE_ZIP	INTEGER	No	No
ID ISM USER	ID_ISM_USER	INTEGER	No	No

BSSM015T
Name: HIER
Code: BSSM015T
Label:
Owner:
Number: 1000
PK constraint:
Source: Entity HIERARCHY

Options

data capture NONE

in BSSMOOCS

{

index in BSSMOOCO

}

Description
This is a hierarchy of all the skills and their groups. Each row will point to its parent row to establish the tree structure for all the skills.

Column List

Name	Code	Type	P	M
ID HIER	ID_HIER	INTEGER	Yes	Yes
CODE DOT	CODE_DOT	CHAR(9)	No	No
CODE SOC	CODE_SOC	CHAR(6)	No	No
ID PARENT HIER	ID_PARENT_HIER	INTEGER	No	Yes
CODE HIER STATUS	CODE_HIER_STATUS	INTEGER	No	Yes
ID CRE	ID_CRE	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
MST CREATED	MST_CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	MST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM016T
Name: HIER SKILL
Code: BSSM016T
Label:
Owner:
Number: 20000
PIK constraint:
Source: Entity HIERARCHY SKILL

Options

data capture NONE in BSSMOOCS

{

index in BSSMOOCO

}

Description
The Title Skill table is an associative table used to resolve the many to many relationship between titles and skills. The table represents skills required for a given title.

Column List

Name	Code	Type	P	M
ID SKILL	ID_SKILL	INTEGER	Yes	Yes
ID HIER	ID_HIER	INTEGER	Yes	Yes
NUM SKILL RANK	NUM_SKILL_RANK	SMALLINT	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	MST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM017T
Name: JO SPC PGMS
Code: BSSM017T
Label:
Owner:
Number: 120000
PIK constraint:
Source:

Options

in BSSM017S

{

index in BSSM0170

}

partitioning key (ID JO

)

Description
This lists all of the special programs that a job order has. It will be used when matching to check the special programs that a job seeker is eligible for.

Column List

Name	Code	Type	P	M
ID JO	ID_JO	INTEGER	Yes	Yes
ID SPCL PGM	ID_SPCL_PGM	INTEGER	Yes	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM018T
Name: IETC OFFICE
Code: BSSM018T
Label:
Owner:
Number: 100
PK constraint:
Source: Entity IETC OFFICE

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Description
IETC Offce Designation..

Column List

Name	Code	Type	P	M
ID OFFICE	ID_OFFICE	SMALLINT	Yes	Yes
CODE REGION	CODE_REGION	INTEGER	No	Yes
CODE ZIP	CODE_ZIP	INTEGER	No	Yes
ID PARENT OFFICE	ID_PARENT_OFFICE	SMALLINT	No	Yes
TEXT OFFICE NUM M	TEXT_OFFICE_NUM	CHAR(4)	No	Yes
NAME IETC OFFICE	NAME_IETC_OFFICE	CHAR(40)	No	Yes
TEXT ADDR1	TEXT_ADDR1	CHAR(35)	No	Yes
TEXT ADDR2	TEXT_ADDR2	CHAR(35)	No	Yes
TEXT CITY	TEXT_CITY_CITY	CHAR(18)	No	Yes
TEXT CITY UP	TEXT_CITY_UP	CHAR(18)	No	Yes
CODE CNTY	COD_CNTY_	SMALLINT	No	Yes
TEXT CNTRY	TEXT_CNTRY	CHAR(25)	No	Yes
TEXT PHN NUM	TEXT_PHN_NUM	CHAR(10)	No	Yes
TEXT PHN EXT	TEXT_PHN_EXT	CHAR(6)	No	Yes
TEXT FAX PHN NUM	TEXT_FAX_PHN_NUM	CHAR(10)	No	Yes
TEXT EMAIL ADDR	TEXT_EMAIL_ADDR	CHAR(40)	No	Yes
CODE PRNT ROUTER	CODE_PRNT_ROUTER	CHAR(5)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM019T
Name: IETC PARTNERS
Code: BSSM01 9T
Label:
Owner:
Number: 100
PIK constraint:
Source: Entity IETC PARTNERS

Options
in BSSMOOCS
{
index in BSSMOOCO

}

Description
IETC Partner table.

Column List

Name	Code	Type	P	M
ID PARTNER	ID_PARTNER	INTEGER	Yes	Yes
TEXT PARTNER	TEXT_PARTNER	CHAR(40)	No	Yes

BSSM020T
Name: JO
Code: BSSM020T
Label:
Owner:
Number: 240000
PK constraint:
Source: Entity JOB ORDER

Options
in BSSM020S
{
index in BSSM0200
}
partitioning key (ID_JO
)

Description
Represents a request for possible employment with a registered employer.

Column List

Name	Code	Type	P	M
ID JO	ID_JO	INTEGER	Yes	Yes
CODE CURRENT STAT	CODE_CURRENT_STAT	INTEGER	No	Yes
ID USER	ID_USER	INTEGER	No	Yes
CODE EDUC	CODE_EDUC	INTEGER	No	Yes
ID SRVC DLVRY AREA	ID_SRVC_DLVRY_AREA	CHAR(2)	No	Yes
CODE PAY UNIT	CODE_PAY_UNIT	INTEGER	No	Yes
DATE JO CLOSE	DATE_JO_CLOSE	DATE	No	Yes
TEXT JOB TITLE	TEXT_JOB_TITLE_TITLE	CHAR(40)	No	Yes
TEXT LOC ADDR1	TEXT_LOC_ADDR1	CHAR(35)	No	Yes
TEXT LOC ADDR2	TEXT_LOC_ADDR2	CHAR(35)	No	Yes
TEXT LOC CITY	TEXT_LOC_CITY	CHAR(18)	No	Yes
TEXT LOC CITY UP	TEXT_LOC_CITY_UP	CHAR(18)	No	Yes
CODE LOC ST	CODE_LOC_ST	CHAR(2)	No	Yes
CODE CNTY	CODE_CNTY	SMALLINT	No	Yes
CODE LOC ZIP	CODE_LOC_ZIP	INTEGER	No	Yes
CODE WRK HRS	CODE_WRK_HRS	INTEGER	No	Yes
FLAG AFFRM ACT	FLAG_AFFRM_ACT	CHAR(1)	No	Yes
NUM JOB OPN	NUM_JOB_OPN	SMALLINT	No	Yes
NUM OF SKILLS	NUM_OF_SKILLS	SMALLINT	No	Yes
FLAG PUBLIC TRANS	FLAG_PUBLIC_TRANS	CHAR(1)	No	Yes
TEXT SALARY RANGE	TEXT_SALARY_RANGE	CHAR(24)	No	Yes

AMT PAY OFFER	AMT_PAY_OFFER	DECIMAL(9,2)	No	Yes
AMT MAX NORM PAY	AMT_MAX_NORM_PAY	DECIMAL(9,2)	No	Yes
CODE TEMP PERM	CODE_TEMP_PERM	INTEGER	No	Yes
CODE WORK TYPE	CODE_WORK_TYPE_TYPE	INTEGER	No	Yes
CODE SOC	CODE_SOC	CHAR(6)	No	Yes
FLAG FIRST SHFT	FLAG_FIRST_SHFT	CHAR(1)	No	Yes
FLAG SECOND SHFT	FLAG_SECOND_SHFT	CHAR(1)	No	Yes
FLAG THIRD SHFT	FLAG_THIRD_SHFT	CHAR(1)	No	Yes
FLAG SPLIT SHIFT	FLAG_SPLIT_SHFT	CHAR(1)	No	Yes
FLAG ROTATING SHFT	FLAG_ROTATING_SHFT	CHAR(1)	No	Yes
ID JO CREATOR	ID_JO_CREATOR	INTEGER	No	Yes
ID JO OWNER	ID_JO_OWNER	INTEGER	No	No
FLAG SPCL PGM	FLAG_SPCL_PGM	CHAR(1)	No	Yes
FLAG DAY MATCH	FLAG_DAY_MATCH	CHAR(1)	No	Yes
FLAG SEND RESUME	FLAG_SEND_RESUME	CHAR(1)	No	Yes
CODE SUP EMP CNT	CODE_SUP_EMP_CNT	INTEGER	No	Yes
CODE SUP JS	CODE_SUP_JS	INTEGER	No	Yes
FLAG SUP JS SEND	FLAG_SUP_JS_SEND	CHAR(1)	No	Yes
FLAG SUP EMP SEND	FLAG_SUP_EMP_SEND	CHAR(1)	No	Yes
FLAG SUP EMP PHN	FLAG_SUP_EMP_PHN	CHAR(1)	No	Yes
FLAG SUP JS PHN	FLAG_SUP_JS_PHN	CHAR(1)	No	Yes
FLAG SUP EMP CNAME	FLAG_SUP_EMP_CNAME	CHAR(1)	No	Yes
FLAG SUP JS CNAME	FLAG_SUP_JS_CNAME	CHAR(1)	No	Yes
FLAG SUP EMP ENAME	FLAG_SUP_EMP_ENAME	CHAR(1)	No	Yes
FLAG SUP JS ENAME	FLAG_SUP_JS	CHAR(1)	No	Yes
FLAG SUP EMP EMAIL	FLAG_SUP_EMP_EMAIL	CHAR(1)	No	Yes
FLAG SUP JS EMAIL	FLAG_SUP_JS_EMAIL	CHAR(1)	No	Yes
FLAG SUP EMP FAX	FLAG_SUP_EMP_FAX	CHAR(1)	No	Yes
FLAG SUP JS FAX	FLAG_SUP_JS_FAX	CHAR(1)	No	Yes
FLAG SEND EMP COMM	FLAG_SEND_EMP_COMM	CHAR(1)	No	Yes
FLAG SHOW MAP	FLAG_SHOW_MAP_MAP	CHAR(1)	No	Yes
TEXT JO IDENTIFIER	TEXT_JO_IDENTIFIER	CHAR(25)	No	Yes
FLAG ALW MAN CLOSE	FLAG_ALW_MAN_CLOSE	CHAR(1)	No	Yes
DATE HOLD UNTIL	DATE_HOLD_UNTIL	DATE	No	No
FLAG NEEDS MATCH	FLAG_NEEDS_MATCH	CHAR(1)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
ID CREATED USER	ID_CREATED_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes
TEXT JS SPCL INST	TEXT_JS_SPCL_INST	VARCHAR(255)	No	Yes
TEXT STAFF NOTES	TEXT_STAFF_NOTES	VARCHAR(1000)	No	Yes
TEXT ADDL INFO	TEXT_ADDL_INFO	VARCHAR(1000)	No	Yes
TEXT EMP SPCL INST	TEXT_EMP_SPCL_INST	VARCHAR(255)	No	Yes
TEXT DESC DUTIES	TEXT_DESC_DUTIES	VARCHAR(1000)	No	Yes

BSSM021T

Name: JO STAT
Code: BSSM021T
Label:
Owner:
Number: 720000
PK constraint:
Source: Entity JOB-ORDER-STATUS

Options

in BSSM021S

{

index in BSSM0210

}

partitioning key (ID_JO

)

Description

Jon Order Status.

Column List

Name	Code	Type	P	M
ID JO	ID_JO	INTEGER	Yes	Yes
TMST BEGIN	TMST_BEGIN	TIMESTAMP	Yes	Yes
CODE STAT	CODE_STAT	INTEGER	No	Yes
TMST END	TMST_END	TIMESTAMP	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM022T

Name: JO SKILL
Code: BSSM022T
Label:
Owner:
Number: 4000000
PK constraint:
Source: Entity JOB-ORDER-SKILL

Options

in BSSM022S

{

index in BSSM0220

}

partitioning key (ID_JO

)

Description

This is the set of skills that are associated with a particular job offer. It will be used heavily in

the matching portion of the system.

Column List

Name	Code	Type	P	M
ID JO	ID_JO	INTEGER	Yes	Yes
ID SKILL	ID_SKILL	INTEGER	Yes	Yes
ID HIER	ID_HIER	INTEGER	No	Yes
ID QUANT	ID_QUANT	INTEGER	No	Yes
NUM QUANT RANK	NUM_QUANT_RANK	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes

BSSM023T

Name: JS
Code: BSSM023T
Label:
Owner:
Number: 3000000
PK constraint:
Source: Entity JOB SEEKER

Options

in BSSM023S

{

index in BSSM0230

}

partitioning key (ID_USER

)

Description
This table contains all of the individuals who are looking for jobs or have in the past. These are the job seekers.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
CODE CURRENT STAT	CODE_CURRENT_STAT	INTEGER	No	Yes
CODE SOC	CODE_SOC	CHAR(6)	No	Yes
CODE VET STAT	CODE_VET_STAT	SMALLINT	No	Yes
CODE ETHNIC	CODE_ETHNIC	INTEGER	No	Yes
CODE EDUC	CODE_EDUC	INTEGER	No	Yes
CODE CNTY	CODE_CNTY	SMALLINT	No	Yes
CODE ST	CODE_ST	CHAR(2)	No	Yes
CODE PAY UNIT	CODE_PAY_UNIT	INTEGER	No	Yes
CODE TRAVEL DIST	CODE_TRAVEL_DIST	INTEGER	No	Yes
ID SSN	ID_SSN	CHAR(9)	No	Yes
NAME FIRST	NAME_FIRST	CHAR(20)	No	Yes

NAME FIRST UP	NAME_FIRST_UP	CHAR(20)	No	Yes
NAME MIDDLE INIT	NAME_MIDDLE_INIT	CHAR(1)	No	Yes
NAME LAST	NAME_LAST	CHAR(40)	No	Yes
NAME LAST UP	NAME_LAST_UP	CHAR(40)	No	Yes
TEXT ADDR1	TEXT_ADDR1	CHAR(35)	No	Yes
TEXT ADDR2	TEXT_ADDR2	CHAR(35)	No	Yes
TEXT CITY	TEXT_CITY	CHAR(18)	No	Yes
TEXT CITY UP	TEXT_CITY_UP	CHAR(18)	No	Yes
CODE ZIP	CODE_ZIP	INTEGER	No	Yes
CODE CNTRY	CODE_CNTRY	CHAR(1)	No	Yes
TEXT PHN NUM	EXT_PHN_NUM	CHAR(10)	No	No
TEXT PHN EXT	EXT_PHN_EXT	CHAR(6)	No	Yes
TEXT WRK PHN NUM	EXT_WRK_PHN_NUM	CHAR(10)	No	Yes
TEXT FAX PHN NUM	TEXT_FAX_PHN_NUM	CHAR(10)	No	No
DATE BIRTH	DATE_BIRTH	DATE	No	No
FLAG TEMP	FLAG_TEMP	CHAR(l)	No	Yes
FLAG PERM	FLAG_PERM	CHAR(1)	No	Yes
CODE GENDER	CODE_GENDER	INTEGER	No	Yes
FLAG SUPPRESS IND	FLAG_SUPPRESS_IND	CHAR(1)	No	Yes
FLAG EMPLMNT STAT	FLAG_EMPLMNT_STAT	CHAR(1)	No	Yes
FLAG SCHOOL STAT	FLAG_SCHOOL_STAT	CHAR(1)	No	Yes
NUM TRAVEL DISTNCE	NUM_TRAVEL_DISTNCE	NUMERIC(3)	No	Yes
FLAG WORK IN USA	FLAG_WORK_IN_USA	CHAR(1)	No	Yes
FLAG TMP AGENCY	FLA_TMP_AGENCY	CHAR(1)	No	Yes
AMT MIN PAY REQ	AMT_MIN_PAY_REQ	DECIMAL(9,2)	No	Yes
MT MIN NORM PAY	AMT_MIN_NORM_PAY	DECIMAL(9,2)	No	Yes
FLAG PART TIME	FLAG_PART_TIME	CHAR(1)	No	Yes
FLAG FULL TIME	FLAG_FULL_TIME_TIME	CHAR(1)	No	Yes
FLAG FIRST SHIFT	FLAG_FIRST_SHFT	CHAR(1)	No	Yes
FLAG SECOND SHFT	FLAG_SECOND_SHFT	CHAR(1)	No	Yes
FLAG THIRD SHFT	FLAG_THIRD_SHFT	CHAR(1)	No	Yes
FLAG ROTATING SHFT	FLAG_ROTATING_SHFT	CHAR(1)	No	Yes
FLAG SPLIT SHFT	FLAG_SPLIT_SHFT	CHAR(1)	No	Yes
TEXT EMAIL ADDR	TEXT_EMAIL_ADDR	CHAR(40)	No	Yes
text email _addr_up	TEXT_EMAIL_ADDR_UP	CHAR(40)	No	Yes
FLAG SPCL PGM	FLAG_SPCL_PGM	CHAR(1)	No	Yes
NUM OF SKILLS	NUM_OF_SKILLS	SMALLINT .	No	Yes
CODE MATCH ZIP	CODE_MATCH_ZIP	INTEGER	No	Yes
FLAG VET ACTV DTY	FLAG_VET_ACTV_DTY	CHAR(1)	No	Yes
FLAG VET VIETNAM	FLAG_VET_VIETNAM	CHAR(1)	No	Yes
FLAG VET SPOUSE	FLAG_VET_SPOUSE	CHAR(1)	No	Yes
FLAG VET HNR DSCHG	FLAG_VET_HNR_DSCHG	CHAR(1)	No	Yes
FLAG VET DSBLTY	FLAG_VET_DSBLTY	CHAR(1)	No	Yes
CODE VET DSBLTY	CODE_VET_DSBLTY	INTEGER	No	Yes
CODE VET BRANCH	CODE_VET_BRANCH	INTEGER	No	Yes
DATE VET FROM	DATE_VET_FROM	DATE	No	No
DATE VET TO	DATE_VET_TO	DATE	No	No
TEXT SCHOOL CODE	TEXT_SCHOOL_CODE	CHAR(2)	No	Yes
FLAG SEND JS COMM	FLAG_SEND_JS_COMM	CHAR(1)	No	Yes
TEXT MOM MADN NAME	TEXT_MOM_MADN_NAME	CHAR(40)	No	Yes
FLAG SUPRSS WH EMP	FLAG_SUPRSS_WH_EMP	CHAR(1)	No	Yes
FLAG SEASONAL WRK	FLAG_SEASONAL_WRK	CHAR(1)	No	Yes
FLAG DISABILITY	FLAG_DISABILITY	CHAR(l)	No	Yes
FLAG NEEDS MATCH	FLAG_NEEDS_MATCH	CHAR(l)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
ID CREATED USER	ID_CREATED_USER	INTEGER	No	Yes

TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM024T
Name: JS SKILL
Code: BSSM024T
Label:
Owner:
Number: 60500000
PK constraint:
Source: Entity JOB SEEKER SKILL

Options

in BSSM024S

{

index in BSSM0240

}

partitioning key (ID_USER

)

Description
Job seekers skills.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
ID SKILL	ID_SKILL	INTEGER	Yes	Yes
ID HIER	ID_HIER	INTEGER	No	Yes
ID QUANT	ID_QUANT	INTEGER	No	Yes
NUM QUANT RANK	NUM_QUANT_RANK	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes

BSSM02ST
Name: JS SPC PGMS
Code: BSSM025T
Label:
Owner:
Number: 60000
PK constraint:
Source: Entity JS SPC PGMS

Options

in BSSM025S
{
index in BSSM0250
}
partitioning key (ID_USER
)

Description
This is the list of special programs that a job seeker is eligible for. It will be used in matching to see if the special programs are also associated with the job.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
ID SPCL PGM	ID_SPCL_PGM	INTEGER	Yes	Yes
TMST EXPIR	TMST_EXPIR	TIMESTAMP	Yes	Yes
ID DELETED USER	ID_DELETED_USER	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM026T
Name: JS SRVC PROVIDED
Code: BSSM026T
Label:
Owner:
Number: 12000000
PK constraint:
Source: Entity JOB SKR SERVICE PROVIDED

Options

in BSSM026S

{

index in BSSM026S

}

partitioning key (ID_JS_USER

)

Description
This is a list of the services that a staff member has performed for a job seeker.

Column List

Name	Code	Type	P	M
ID SRVCS PRVDED	ID_SRVCS_PRVDED	INTEGER	Yes	Yes
ID JS USER	ID_JS_USER	INTEGER	Yes	Yes

ID SRVC	ID_SRVC	INTEGER	No	Yes
ID USER	ID_USER	INTEGER	No	No
DATE SRVC	DATE_SRVC	DATE	No	Yes
FLAG DELETE	FLAG_DELETE	CHAR(1)	No	Yes
ID UI ACCT NUM	ID_UI_ACCT_NUM	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes
TEXT COMMENT	TEXT_COMMENT	VARCHAR(255)	No	Yes

BSSM027T
Name: JS SRVCS
Code: BSSM027T
Label:
Owner:
Number: 100
PK constraint:
Source: Entity JOBSKR SERVICES

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Description
This is a list of all possible services that a staff can perform for a job seeker.

Column List

Name	Code	Type	P	M
ID SRVC	ID_SRVC	INTEGER	Yes	Yes
ID SRVCS TCDE	ID_SRVCS_TCDE	INTEGER	No	Yes
TEXT SRVC	TEXT_SRVC	CHAR(55)	No	Yes
FLAG ISM SRVCS	FLAG_ISM_SRVCS	CHAR(1)	No	Yes
FLAG OBTAIN EMP	FLAG_OBTAIN_EMP	CHAR(1)	No	Yes
FLAG RPT TO ENDS	FLAG_RPT_TO_ENDS	CHAR(1)	No	Yes
ID SRVCS XREF	ID_SRVCS_XREF	INTEGER	No	Yes
CODE JS CAT	CODE_JS_CAT	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Ye
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM028T
Name: JS STAT
Code: BSSM028T
Label:
Owner:
Number: 9000000
PK constraint:
Source: Entity JOB-SEEKER-STATUS

Options

in BSSM028S

{

index in BSSM028S

}

partitioning key (ID_USER

)

Description
This is the status of the job seeker. It keeps track of the different statuses over time.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
TMST BEGIN	TMST_BEGIN	TIMESTAMP	Yes	Yes
CODE STAT	CODE_STAT	INTEGER	No	Yes
TMST END	TMST_END	TIMESTAMP	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM029T
Name: JS WRK HSTRY
Code: BSSM029T
Label:
Owner:
Number: 9000000
PK constraint:
Source: Entity JOB SEEKER WORK HISTORY

Options

in BSSM029S

{

index in BSSM0290

}

partitioning key (ID_USER

)

Description
This is the job seeker's work history of jobs and employers.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
ID WORK HIST	ID_WORK_HIST	INTEGER	Yes	Yes
NAME EMP	NAME_EM_P	CHAR(30)	No	Yes
TEXT CITY Y	TEXT_CITY	CHAR(18)	No	Yes
TEXT ST ABBREV	TEXT_ST_ABBREV	CHAR(2)	No	Yes
TEXT CNTRY	TEXT_CNTRY	CHAR(25)	No	Yes
TEXT TITLE	TEXT_TITLE	CHAR(40)	No	Yes
TEXT DATE FROM	TEXT_DATE_FROM	CHAR(110)	No	Yes
TEXT DATE TO O	TEXT_DATE_TO	CHAR(10)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM030T

Name: PROGRAM GROUP
Code: BSSM030T
Label:
Owner:
Number: 250
PIK constraint:
Source:

Options

in BSSMOOCS
{
index in BSSMOOCO
}

Column List

Name	Code	Type	P	M
CODE GRP	CODE_GRP	CHAR(8)	Yes	Yes
FLAG ENCRYPT IN	FLAG_ENCRYPT_IN	CHAR(1)	No	Yes
FLAG ENCRYPT OUT	FLAG_ENCRYPT_OUT	CHAR(1)	No	Yes

BSSM031T

Name: PROGRAM GROUP NAV
Code: BSSM031T
Label:
Owner:
Number: 1000
PIK constraint:
Source:

Options

in BSSMOOCS

{

index in BSSMOOCO

}

Column List

Name	Code	Type	P	M
CODE FROM GROUP	CODE_FROM_GROUP	CHARACTER(8)	Yes	Yes
CODE TO GROUP	CODE_TO_GROUP	CHARACTER(8)	Yes	Yes
ID RIGHT	ID_RIGHT	INTEGER	Yes	Yes

BSSM032T

Name: SESSION SKILL
Code: BSSM032T
Label:
Owner:
Number: 10000
PIK constraint:
Source:

Options

in BSSM032S

{

index in BSSM0320

}

Description

This is a set of skills that a job seeker or an employer contact is working on during a particular session. They will ultimately be written to the corresponding skill table for the job seeker or the job order.

Column List

Name	Code	Type	P	M
ID SESSION	ID_SESSION	CHAR(16)	Yes	Yes
ID SKILL	ID_SKILL	INTEGER	Yes	Yes
ID HIER	ID_HIER	INTEGER	No	Yes

ID QUANT	ID_QUANT	INTEGER	No	Yes
NUM QUANT RANK	NUM_QUANT_RANK	INTEGER	No	Yes
NUM JS SKILL CNT	NUM_JS_SKILL_CNT	INTEGER	No	Yes
NUM MTCH SKILL CNT	NUM_MTCH_SKILL_CNT	INTEGER	No	Yes

BSSM033T

Name: CODE LOOKUP HDR
Code: BSSM033T
Label: Code Lookup Hdr
Owner:
Number: 200
PK constraint:
Source:

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Description
This is the header table for the code look ups. It will have the table number and a description for all code tables that are part of the child table.

Column List

Name	Code	Type	P	M
ID LOOKUP TBL	ID_LOOKUP_TBL	SMALLINT	Yes	Yes
TEXT COLUMN	TEXT_COLUMN	CHAR(18)	No	Yes
TEXT LOOKUP P	TEXT_LOOKUP	CHAR(30)	No	Yes
FLAG CHAR REQD	FLAG_CHAR_REQD	CHAR(1)	No	Yes

BSSM034T

Name: USER RIGHTS
Code: BSSM034T
Label:
Owner:
Number: 3075000
PK constraint:
Source:

Options
in BSSM034S
{
index in BSSM0340
}

Description
This table contains all of the rights for a user. These indicate what the user is allowed to do in the system.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
ID RIGHT	ID_RIGHT	INTEGER	Yes	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM03ST
Name: MAIL FORM
Code: BSSM035T
Label:
Owner:
Number: 20
PIK constraint:
Source: Entity MAIL FORM

Options
data capture NONE
in BSSMOOCS
{
index in BSSMOOCO
}

Column List

Name	Code	Type	P	M
ID FORM	ID_FORM	INTEGER	Yes	Yes
TEXT SUBJECT	TEXT_SUBJECT	CHAR(80)	No	Yes
NUM LIST LENGTH	NUM_LIST_LENGTH	SMALLINT	No	Yes
ID ISM USER	ID _ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM036T
Name: MAIL FORM LO
Code: BSSM036T
Label:
Owner:
Number: 2000

PK constraint:
Source: Entity MAIL_FORM_LO

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Column List

Name	Code	Type	P	M
ID FORM	ID_FORM	INTEGER	Yes	Yes
ID OFFICE	ID_OFFICE	SMALLINT	Yes	Yes
ID ISM USER DEFLT	ID_ISM_USER_DEFLT	INTEGER	No	Yes
TEXT TITLE	TEXT_TITLE	CHAR(40)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM037T
Name: OCCUP SKILL
Code: BSSM037T
Label:
Owner:
Number: 15000
PIK constraint:
Source: Entity OCCUPATIONAL_SKILL

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Description
The occupational skill names a capability which is used when performing work named by a Occupational Title.

Column List

Name	Code	Type	P	M
ID SKILL	ID_SKILL	INTEGER	Yes	Yes
ID SKILL TYPE	ID_SKILL_TYPE	INTEGER	No	Yes
CODE OS STATUS	CODE_OS_STATUS	INTEGER	No	Yes
CODE REQUEST SRC	CODE_REQUEST_SRC	INTEGER	No	Yes
ID SKILL REPLACE	ID_SKILL_REPLACE	INTEGER	No	Yes
ID BATCH UPDATE	ID_BATCH_UPDATE	CHAR(8)	No	Yes
TMST BATCH UPDATE	TMST_BATCH_UPDATE	TIMESTAMP	No	Yes
TMST CREATED	TMST_CREATED	IMESTAMP	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	IMESTAMP	No	Yes

BSSM039T

Name: OFFICE_ZIPS

Code: BSSM039T

Label:
Owner:
Number: 1000
PK constraint:
Source: Entity LOCAL_OFFICE_ZIPS

Options
in BSSMOOCS
{
index in BSSM00C0
}

Description
Zip codes assigned to each office.

Column List

Name	Code	Type	P	M
ID OFFICE	ID_OFFICE	SMALLINT	Yes	Yes
CODE ZIP	CODE_ZIP	INTEGER	Yes	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM040T

Name: PARTNER OFFICE
Code: BSSM040T
Label:
Owner:
Number: 100
PK constraint:
Source:

Options
in BSSM00CS
{
index in BSSM00C0
}

Description
This is an associative table that relates a partner to their associated IETC offices.

Column List

Name	Code	Type	P	M
ID PARTNER	ID_PARTNER	INTEGER	Yes	Yes

ID OFFICE	ID_OFFICE	SMALLINT	Yes	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM041T

Name: PHONE MESSAGE
Code: BSSM041T
Label:
Owner:
Number: 500
PIK constraint:
Source: Entity PHONE MESSAGE

Options
in BSSM00CS
{
index in BSSM00C0
}

Column List

Name	Code	Type	P	M
ID PHN MSG	ID_PHN_MSG	INTEGER	Yes	Yes
TEXT PHN MSG	TEXT_PHN_MSG	CHAR(50)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM042T

Name: PHONE MESSAGE LO
Code: BSSM042T
Label:
Owner:
Number: 500
PIK constraint:
Source: Entity PHONE_MESSAGE_L0

Options
in BSSM00CS
{
index in BSSM00C0
}

Column List

Name	Code	Type	P	M
ID PHN MSG	ID_PHN_MSG	INTEGER	Yes	Yes
ID OFFICE	ID_0FFFCE	SMALLINT	Yes	Yes
ID PNS	ID_PNS	NUMERIC(10)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM043T

Name: QUAL CANDIDATE
Code: BSSM043T

Label:
Owner:
Number: 7300000
PIK constraint:
Source: Entity QUALIFIED_CANDIDATE

Options
in BSSM043S
{
index in BSSM0430
}
partitioning key (ID_JO)

Description
Qualified candidate represents an applicant whose qualifications meet or exceeded the stated requirements of a job order.

Column List

Name	Code	Type	P	M
ID JO	ID_JO	INTEGER	Yes	Yes
ID USER	ID_USER	INTEGER	Yes	Yes
FLAG SEND JS COMM	FLAG_SEND_JS_COMM	CHAR(1)	No	Yes
FLAG SEND EMP COMM	FLAG_SEND_EMP_COMM	CHAR(1)	No	Yes
TMST LAST VIEWED	TMST_LAST_VIEWED	TIMESTAMP	No	No
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes

BSSM044T
Name: REASON CODE
Code: BSSM044T
Label:
Owner:
Number: 200
PIK constraint:
Source: Entity REASON_CODE

Options
in
BSSM00CS
{
index in
BSSM00C0
}

Column List

Name	Code	Type	P	M
ID REASON	ID_REASON	SMALLINT	Yes	Yes
ID PHN MSG	ID_PHN_MSG	INTEGER	No	Yes
ID GROUP	ID_GROUP	SMALLINT	No	Yes

TEXT REASON	TEXT_REASON	CHAR(50)	No	Yes
FLAG MASS CALL	FLAG_MASS_CALL	CHAR(1)	No	Yes
ID EMAIL FORM	ID_EMAIL_FORM	INTEGER	No	Yes
ID LETTER FORM	ID_LETTER_FORM	INTEGER	No	Yes
CODE PRIORITY	CODE_PRIORITY	CHAR(1)	No	Yes
TEXT SP NAME	TEXT_SP_NAME	CHAR(8)	No	Yes
TMST LAST RUN	TMST_LAST_RUN	TIMESTAMP	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM046T

Name: RFRL ACTION
Code: BSSM046T
Label:
Owner:
Number: 3000000
PK constraint:
Source: Entity REFERRAL_ACTION

Options

in BSSM046S
{
index in
BSSM0460
}
partitioning key (ID_JO)

Description

This is an output of the matching process. Any qualified candidates that are viewed are considered referred and need to have referral data track on them.

Column List

Name	Code	Type	P	M
ID JO	ID_JO	INTEGER	Yes	Yes
ID USER	ID_USER	INTEGER	Yes	Yes
CODE REFER RESULT	CODE_REFER_RESULT	INTEGER	No	Yes
FLAG REFER BY EMP	FLAG_REFER_BY EMP	CHAR(1)	No	Yes
FLAG REFER	FLAG_REFER_	CHAR(1)	No	Yes
FLAG MANUAL CNTC	FLAG_MANUAL_CNTC	CHAR(1)	No	Yes
FLAG SEND JS COMM	FLAG_SEND_JS_COMM	CHAR(1)	No	Yes
FLAG SEND EMP COMM	FLAG_SEND_EMP_COMM	CHAR(1)	No	Yes
ID REFER RSLT MOD	ID_REFER_RSLT_MOD	INTEGER	No	No
TMST RFR RSLT MOD	TMST_RFR_RSLT_MOD	TIMESTAMP	No	No
DATE LAST EARNINGS	DATE_LAST_EARNINGS	DATE	No	No
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATETIMESTAMP		No	Yes

BSSM048T

Name: SEQ NUM
Code: BSSM048T
Label:
Owner:
Number: 100
PIK constraint:
Source:

Options
in BSSM00CS
{
index in BSSM00C0
}

Description
This table contains the next sequence number for all of the sequentially assigned key fields. It is used to assign the value for the keys.

Column List

Name	Code	Type	P	M
TEXT TABLE NAME	TEXT_TABLE_NAME	CHAR(8)	Yes	Yes
NUM SEQ NUM	NUM_SEQ_NUM	INTEGER	No	Yes

BSSM049T

Name: SIC CODE
Code: BSSM049T
Label:
Owner:
Number: 40000
PIK constraint:
Source: Entity SIC_CODE

Options
data capture
NONE
in BSSM049S
{
index in
BSSM0490
}

Description

This is a list of all of the standard industrial classifications (SIC) for employers. The table also indicates if the sic is for a temporary agency.

Column List

Name	Code	Type	P	M
CODE SIC	CODE_SIC	CHAR(5)	Yes	Yes
TEXT SIC	TEXT_SIC	CHAR(75)	No	Yes
FLAG TMP AGENCY	FLAG_TMP_AGENCY	CHAR(1)	No	Yes

BSSM050T

Name: SKILL QUANT
Code: BSSM050T
Label:
Owner:
Number: 20
PK constraint:
Source: Entity SKILL_QUANTIFIER

Options
in
BSSM00CS
{
index in
BSSM00C0
}

Description
This table holds the possible quantifiers for a skill type.

Column List

Name	Code	Type	P	M
ID QUANT	ID_QUANT	INTEGER	Yes	Yes
ID SKILL TYPE	ID_SKILL_TYPE	INTEGER	No	Yes
TEXT QUANT	TEXT_QUANT	CHAR(20)	No	Yes
NUM QUANT RANK	NUM_QUANT_RANK	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM051T

Name: SKILL QUANT TYPE
Code: BSSM051T
Label:
Owner:
Number: 20
PK constraint:
Source: Entity SKILL_TYPE

Options
in

BSSM00CS
{
index in
BSSM00C0
}

Description
This table has all of the different skill types to categorize the various skills so that the specifics of any quantifiers can be assigned to a type.

Column List

Name	Code	Type	P	M
ID SKILL TYPE	ID_SKILL_TYPE	INTEGER	Yes	Yes
TEXT SKILL TYPE	TEXT_SKILL_TYPE	CHAR(20)	No	Yes
NUM TYPE RANK	NUM_TYPE_RANK	SMALLINT	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM053T

Name: SPCL PGMS
Code: BSSM053T
Label:
Owner:
Number: 20
PK constraint:
Source: Entity SPECIAL PROGRAMS

Options
in
BSSM00CS
{
index in
BSSM00C0
}

Description
Special programs offered by potential postionee/positionor system partners to employ the unemployed.

Column List

Name	Code	Type	P	M
ID SPCL PGM	ID_SPCL_PGM	INTEGER	Yes	Yes
ID PARTNER	ID_PARTNER	INTEGER	No	Yes
TEXT PGM	TEXT_PGM	CHAR(50)	No	Yes
TMST EFF	TMST_EFF	TIMESTAMP	No	Yes
TMST END	TMST_END	TIMESTAMP	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM054T

Name: SRVC DLVRY AREA
Code: BSSM054T
Label:
Owner:
Number: 30
PIK constraint:
Source: Entity SERVICE_DELIVERY_AREA

Options

in BSSM00CS
{
index in
BSSM00C0
}

Description

Geographical/Political boundary denoting an area in which services are delivered.

Column List

Name	Code	Type	P	M
ID SRVC DLVRY AREA	ID_SRVC_DLVRY_AREA	CHAR(2)	Yes	Yes
ID OFFICE	ID_OFFICE	SMALLINT	No	Yes
TEXT SDA	TEXT_SDA	CHAR(40)	No	Yes
ID ISM USER	ID_ISIM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST UPDATE	TIMESTAMP	No	Yes

BSSM055T

Name: ST
Code: BSSM055T
Label:
Owner:
Number: 50
PK constraint:
Source: Entity STATE

Options

in BSSM00CS
{
index in
BSSM00C0
}

Description

State table.

Column List

Name	Code	Type	P	M
CODE ST	CODE_ST	CHAR(2)	Yes	Yes
TEXT ST NAME	TEXT_ST_NAME	CHAR (20)	No	Yes

BSSM056T

Name: STAFF
Code: BSSM056T
Label:
Owner:
Number: 1500
PK constraint:
Source: Entity STAFF

Options
in
BSSM056S
{
index in
BSSM0560
}

Description
This table contains the information about all of the users who are staff members.

Column List

Name	Code	Type	P	M
ID_USER	ID_USER	INTEGER	Yes	Yes
NAME FIRST	NAME_FIRST	CHAR(20)	No	Yes
NAME FIRST UP	NAME_FIRST_UP	CHAR(20)	No	Yes
NAME MIDDLE INIT	NAME_MIDDLE INIT	CHAR(1)	No	Yes
NAME LAST	NAME_LAST	CHAR(40)	No	Yes
NAME LAST UP	NAME_LAST_UP	CHAR(40)	No	Yes
TEXT EMAIL ADDR	TEXT_EMAIL_ADDR	CHAR(40)	No	Yes
TEXT EMAIL ADDR UP	TEXT_EMAIL_ADDR_UP	CHAR(40)	No	Yes
TEXT PHN NUM	TEXT_PHN_NUM	CHAR(10)	No	Yes
TEXT PHN EXT	TEXT_PHN_EXT	CHAR(6)	No	Yes
TEXT ENDS DESK NUM	TEXT_ENDS_DESK_NUM	CHAR(4)	No	Yes
CODE VET STAFF	CODE_VET_STAFF	INTEGER	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM057T
Name: USER HISTORY
Code: BSSM057T
Label:
Owner:
Number: **3075000**
PK constraint:
Source: Entity STAFF_HISTORY

Options
in BSSM057S
{
index in
BSSM0570
}

Description

This details which office the user is associated with for a certain time period.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
TMST EFF	TMST_EFF	TIMESTAMP	Yes	Yes
TMST END	TMST_END	TIMESTAMP	No	Yes
ID OFFICE	ID OFFICE	SMALLINT	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM058T

Name: UI CLMNT HSTRY
Code: BSSM058T
Label:
Owner:
Number: 200000
PIK constraint:
Source: Entity UI_CLAIMANT_HISTORY

Options

in BSSM058S
{
index in
BSSM0580
}

Description

This details all unemployment claims made by the job seekers.

Column List

Name	Code	Type	P	M
ID SSN	ID_SSN	CHAR(9)	Yes	Yes
TMST BEGIN	TMST_BEGIN	TIMESTAMP	Yes	Yes
TMST END	TMST_END	TIMESTAMP	No	Yes
ID UI FILING OFFICE	ID_UI_FILING_OFFC	SMALLINT	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST LAST UPDATE	TMST LAST UPDATE	TIMESTAMP	No	Yes

BSSM059T

Name: USER LOGIN
Code: BSSM059T
Label:
Owner:
Number: 3075000
PK constraint:
Source: Entity USER_LOGIN

Options

in
BSSM059S
{
index in
BSSM0590
}

Description
This stores the user's id and related information to allow access to the system.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
ID USER TYPE	USER_TYPE_ID	SMALLINT	No	Yes
ID PARTNER	ID_PARTNER	INTEGER	No	Yes
FLAG ENABLED	FLAG_ENABLED	CHAR(1)	No	Yes
TEXT USERNAME LO	TEXT_USERNAME_LO	CHAR(12)	No	Yes
TEXT BASE USER LO	TEXT_BASE_USER_LO	CHAR(7)	No	Yes
NUM SEQ USER NAME	NUM_SEQ_USER_NAME	SMALLINT	No	Yes
TEXT PASSWORD	TEXT_PASSWORD	CHAR(15)	No	Yes
FLAG LOGIN STAT	FLAG_LOGIN_STAT	CHAR(1)	No	Yes
TMST PSWD EXPIRES	TMST_PSWD_EXPIRES	TIMESTAMP	No	Yes
NUM LOGIN ATTEMPTS	NUM_LOGIN_ATTEMPTS	SMALLINT	No	Yes
TMST UNSUC LOGIN	TMST_UNSUC_LOGIN	TIMESTAMP	No	Yes
TMST LAST LOGIN	TMST_LAST_LOGIN	TIMESTAMP	No	Yes
CODE DISABLE RSN	CODE_DISABLE_RSN	INTEGER	No	Yes
NAME FIRST UP	NAME_FIRST_UP	CHAR(20)	No	Yes
NAME LAST UP	NAME_LAST_UP	CHAR(40)	No	Yes
NAME MIDDLE INIT	NAME_MIDDLE_INIT	CHAR(1)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM060T

Name: USER TYPES
Code: BSSM060T
Label:
Owner:
Number: 10
PIK constraint:
Source: Entity USER TYPES

Options
in
BSSMOOCS
{
index in
BSSMOOCO
}

Description

Table containing the types of users.

Column List

Name	Code	Type	P	M
ID USER TYPE	USER_TYPE_ID	SMALLINT	Yes	Yes
TEXT USER TYPE	TEXT_USER_TYPE	CHAR(20)	No	Yes
ID RIGHT	ID_RIGHT	INTEGER	No	Yes
NUM SSN EXPR MINS	NUM_SSN_EXPR_MINS	SMALLINT	No	Yes
NUM MAX LOGINS	NUM_MAX_LOGINS	SMALLINT	No	Yes
NUM PSWD RETRY CNT	NUM_PSWD_RETRY_CNT	SMALLINT	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
CODE USER HOME GRP	CODE_USER_HOME_GRP	CHAR(8)	No	Yes
FLAG REQ CERT AUTH	FLAG_REQ_CERT_AUTH	CHAR(1)	No	Yes
MST CREATED	MST_CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	MST_LAST_UPDATE	IMESTAMP	No	Yes
TEXT LOGON MSG	TEXT_LOGON_MSG	VARCHAR(300)	No	Yes

BSSM061T

Name: VET STAT
Code: BSSM061T
Label:
Owner:
Number: 10
PK constraint:
Source: Entity VET STATUS

Options
in BSSMOOCSS
{
index in
BSSMOOCO
}
Description
This is the code table for veterans status.

Column List

Name	Code	Type	P	M
CODE VET STAT	CODE_VET_STAT	SMALLINT	Yes	Yes
TEXT VET STAT	TEXT_VET_STAT	CHAR(40)	No	Yes
NUM VET STAT RANK	NUM_VET_STAT_RANK	SMALLINT	No	Yes

BSSM062T

Name: WELFARE HSTRY
Code: BSSM062T
Label:
Owner:
Number: 400000
PIK constraint:
Source: Entity WELFARE HISTORY

Options
in

BSSM062S
{
index in
BSSM0620
}

Description
This table keeps a history of any welfare programs that the job seeker participated in.

Column List

Name	Code	Type	P	M
ID SSN	ID SSN	CHAR(9)	Yes	Yes
CODE WELFARE TYPE	CODE_WELFARE_TYPE	INTEGER	Yes	Yes
DATE BEGIN	DATE-BEGIN	DATE	Yes	Yes
DATE END	DATE-END	DATE	No	Yes
FLAG REGISTER WORK	FLAG REGISTER WORK	CHAR(1)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
MST CREATED	MST-CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	MST LAST UPDATE	TIMESTAMP	No	Yes

BSSM063T

Name: ZIP CODE
Code: **BSSM063T**
Label:
Owner:
Number: 7000
PIK constraint:
Source: Entity ZIP_CODE

Options
in
BSSMOOCS
{
index in
BSSMOOCO
}

Description
The collection of zip codes.

Column List

Name	Code	Type	P	M
CODE ZIP	CODE_ZIP	INTEGER	Yes	Yes
NUM ZIP LATITUDE	NUM_ZIP_LATITUDE	DECIMAL(10,4)	No	Yes
NUM ZIP LONGITUDE	NUM_ZIP_LONGITUDE	DECIMAL(10,4)	No	Yes

BSSM065T

Name: ZIP PROXIMITY
Code: BSSM065T
Label:
Owner:
Number: 2550000
PIK constraint:

Source: Entity ZIP PROXIMITY

Options
in
BSSM065S
{
index in
BSSM0650
}

Description
A list of zip codes and their distance to adjacent zip codes.

Column List

Name	Code	Type	P	M
CODE ZIP CODE FROM	CODE_ZIP_CODE_FROM	INTEGER	Yes	Yes
CODE ZIP CODE TO	CODE_ZIP_CODE_TO	INTEGER	Yes	Yes
NUM DISTANCE	NUM_DISTANCE	NUMERIC(7,2)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
MST CREATED	MST_CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	MST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM066T
Name: USER_COMM_MODE
Code: BSSM066T
Label:
Owner:
Number: 30
PIK constraint:
Source:

Options
in
BSSM00CS
{
index in
BSSM00C0
}

Description
Contains the communication method (email, phone, US mail) and priority of each for a given user type.

Column List

Name	**Code**	**Type**	**P**	**M**
ID USER TYPE	ID_USER_TYPE	SMALLINT	Yes	Yes
CODE COMM MODE	CODE_COMM_MODE	INTEGER	Yes	Yes
NUM PRIORITY	NUM_PRIORITY	SMALLINT	No	Yes

BSSM067T
Name: JS EDUCATION
Code: BSSM067T

Label:
Owner:
Number: 9000000
PIK constraint:
Source:

Options
in BSSM067S
{
index in BSSM0670
}
partitioning key (
ID_USER)

Description
Job Seekers Educational History.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
ID EDUCATION	ID_EDUCATION	INTEGER	Yes	Yes
TEXT SCHOOL	TEXT_SCHOOL	CHAR(75)	No	Yes
TEXT YRS ATTENDED	TEXT_YRS_ATTENDED	CHAR(5)	No	Yes
TEXT MAJOR	TEXT_MAJOR	CHAR(60)	No	Yes
TEXT MINOR R	TEXT_MINOR	CHAR(60)	No	Yes
TEXT DEGREE	TEXT_DEGREE	CHAR(60)	No	Yes
TEXT CITY Y	TEXT_CITY	CHAR(18)	No	Yes
TEXT ST ABBREV	TEXT_ST_ABBREV	CHAR(2)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE -	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM068T
Name: PROGRAM YEAR
Code: BSSM068T
Label:
Owner:
Number: 1
PK constraint:
Source:

Options

in
BSSMOOCS
{
index in
BSSMOOCO
}

Column List

Name	Code	Type	P	M
DATE BEGIN PGM YR	DATE_BEGIN_PGM_YR	DATE	Yes	Yes
DATE END PGM YR	DATE_END_PGM_YR	DATE	Yes	Yes

BSSM070T

Name: BFS FORMS
Code: BSSM070T
Label:
Owner:
Number: 10
PIK constraint:
Source:

Options

in
BSSM00CS
{
index in
BSSM00C0
}

Column List

Name	Code	Type	P	M
ID FORM	ID_FORM	INTEGER	Yes	Yes
NUM LINE SEQ	NUM_LINE_SEQ	SMALLINT	Yes	Yes
CODE MERGE	CODE_MERGE	CHAR(1)	No	Yes
MST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
MST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes
TEXT BODY	TEXT_BODY	VARCHAR(80)	No	Yes

BSSM072T

Name: RIGHTS
Code: BSSM072T
Label:
Owner:
Number: 5
PK constraint:
Source:

Options

in BSSM00CS
{
index in BSSM00C0
}

Description
This table contains the list of possible rights that a user can have.

Column List

Name	Code	Type	P	M
ID RIGHT	ID_RIGHT	INTEGER	Yes	Yes
TEXT RIGHT	TEXT_RIGHT	CHAR(80)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM076T

Name: PROCESS CONTROL
Code: BSSM076T
Label:
Owner:
Number: 4000
PK constraint:
Source:

Options
in
BSSM076S
{
index in
BSSM0760
}

Description
This table stores both the information necessary to handle restarting a program and performance statistics from each run of the program. Each job that is run will appear on this table. It is updated through the program when a checkpoint is taken and at the end of the program to set the final counts.

Column List

Name	Code	Type	P	M
ID PROCESS	ID_PROCESS	INTEGER	Yes	Yes
ID BATCH FREQ	ID_BATCH_FREQ	INTEGER	No	Yes
TEXT JOB NAME	EXT_JOB_NAME	CHAR(10)	No	Yes
NUM STEP	NUM_STEP	SMALLINT	No	Yes
CODE PROC STATUS	CODE_PROC_STATUS	INTEGER	No	Yes
TMST START	TMST_START	TIMESTAMP	No	No
TMST COMPLETE	TMST-COMPLETE	TIMESTAMP	No	No
TMST RESTART	TMST_RESTART	TIMESTAMP	No	No
TMST_CHECKPOINT	TMST_CHECKPOINT	TIMESTAMP	No	No
NUM CHKP LUW INT	NUM_CHKP_LUW_INT	INTEGER	No	Yes
NUM CHKP TIME MAX	NUM_CHKP_TIME_MAX	DECIMAL(6)	No	Yes
TMST TERMINATION	TMST_TERMINATION	TIMESTAMP	No	Yes
TEXT CHKP SAVE	TEXT_CHKP_SAVE	VARCHAR(3000)	No	Yes

BSSM077T

Name: PROCESS FILE

Code: BSSM077T

Label:

Owner:

Number: 6000

PIK constraint:

Source:

Options

in BSSM077S

{

index in BSSM0770

}

Description

This contains the information about each file used byh the program to be able to reposition that file at the point of the last commit.

Column List

Name	Code	Type	P	M
ID PROCESS	ID_PROCESS	INTEGER	Yes	Yes
ID FILE	ID_FILE	CHAR(1)	Yes	Yes
TEXT FILE NAME	TEXT_FILE_NAME	VARCHAR(255)	No	Yes
NUM RECS PROCESSED	NUM_RECS_PROCESSED	INTEGER	No	Yes

BSSM086T

Name: BATCH FREQUENCY
Code: BSSM086T
Label: Batch Frequency Table
Owner:
Number: 10
PK constraint:
Source:

Options

in BSSMOOCS
{
index in BSSMOOCO
}

Description

This table contains the different job frequencies (daily, weekly, etc.) and what date they are running for.

Column List

Name	Code	Type	P	M	
ID BATCH FREQ	ID_BATCH_FREQ	INTEGER		Yes	Yes
TEXT BATCH FREQ	TEXT_BATCH_FREQ	CHAR(35)		No	Yes
TMST EFFECTIVE	TMST_EFFECTIVE	TIMESTAMP		No	Yes
TMST LAST EFF	TMST_LAST_EFF	TIMESTAMP		No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP		No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP		No	Yes

BSSM088T

Name: JO BENEFITS
Code: BSSM088T
Label: JO Benefit Table
Owner:
Number: 1200000
PK constraint:
Source:

Options

in
BSSM088S
{
index in
BSSM0880
}
partitioning key (
ID_JO
)

Description

This is the list of benefits that an employer has selected that are applicable for this job offer.

Column List

Name	Code	Type	P	M
ID JO	ID_JO	INTEGER	Yes	Yes
CODE BENEFIT	CODE_BENEFIT	INTEGER	Yes	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATEDT	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM089T

Name: JS CASE MGR
Code: BSSM089T
Label: JS CASE MGR
Owner:
Number: 300000
PK constraint:

Source:

Options

in BSSM089S

{

index in BSSM0890

}

partitioning key (ID_USER)

Description

This table contains the case managers that are working with the job seeker.

Column List

Name	Code	Type	P	M
ID USER	ID_USER	INTEGER	Yes	Yes
ID USER MGR	ID_USER_MGR	INTEGER	Yes	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST CREATED	TMST_CREATED	TIMESTAMP	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM090T

Name: CODE LOOKUP DATA
Code: BSSM090T
Label: Code Lookup Data

Owner:

Number: 2000

PK constraint:

Source:

Options

in BSSM00CS

{

index in BSSM00C0

}

Description

This contains the code (a numeric value) and its description. For some tables, the character value will also be filled in for use in reporting to other systems.

Column List

Name	Code	Type	P	M
ID LOOKUP TBL	ID_LOOKUP_TBL	SMALLINT	Yes	Yes
ID CODE	ID_CODE	INTEGER	Yes	Yes
TEXT LOOKUP	TEXT_LOOKUP	CHAR(40)	No	Yes
FLAG ACTIVE	FLAG_ACTIVE	CHAR(1)	No	Yes
TEXT CHAR CODE	TEXT_CHAR_CODE	CHAR(2)	No	Yes
ID ISM USER	ID_ISM_USER	INTEGER	No	Yes
TMST LAST UPDATE	TMST_LAST_UPDATE	TIMESTAMP	No	Yes

BSSM091T
Name: SOC CODE
Code: BSSM091T
Label: SOC CODE DESCRIPTION
Owner:
Number: 1000
PK constraint:
Source:

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Column List

Name	Code	Type	P	M
CODE SOC	CODE_SOC	CHAR(6)	Yes	Yes
TEXT SOC DESC	TEXT_SOC_DESC	CHAR(120)	No	Yes

BSSM092T
Name: DOT CODE
Code: BSSM092T
Label: DOT CODE DESCRIPTION
Owner:
Number: 1000
PK constraint:
Source:

Options
in BSSMOOCS
{
index in BSSMOOCO
}

Description
This table stores the DOT codes and their descriptions.

Column List

Name	Code	Type	P	M
CODE DOT	CODE_DOT	CHAR(9)	Yes	Yes
TEXT DOT DESC	TEXT_DOT_DESC	CHAR(120)	No	Yes

BSSM093T

Name: SDA ZIP CODE
Code: BSSM093T
Label:
Owner:
Number: 100
PK constraint:
Source:

Options

in BSSM00CS
{
index in BSSM00C0
}

Description

This table stores zip codes that have special SDA's. If the SDA is not found at the county level, the zip code of the location will be used to find it in this table.

Column List

Name	Code	Type	P	M
CODE ZIP	CODE_ZIP	INTEGER	Yes	Yes
ID SRVC DLVRY AREA	ID_SRVC_DLVRY_AREA	CHAR(2)	No	Yes

BSSM094T

Name: DUP SERVLET
TMST
Code: BSSM094T
Label:
Owner:
Number: 1000
PK constraint:
Source:

Options

in BSSM094S
{

index in BSSM0940
}

Description

This table stores the timestamp of the last time a servelet was run by a session (user).

Column List

Name	Code	Type	P	M
ID SERVLET	ID_SERVLET	CHAR(8)	Yes	Yes
ID SESSION	ID_SESSION	CHAR(16)	Yes	Yes
CODE SERVLET PROC	CODE_SERVLET_PROC	CHAR(1)	No	Yes
TMST LAST SERVLET	TMST_LAST_SERVLET	CHAR(17)	No	Yes
TMST LAST JSP	TMST_LAST_JSP	CHAR(17)	No	No
TMST INIT SERVLET	TMST_INIT_SERVLET	CHAR(17)	No	Yes

BSSM095T

Name:

CTIVE USER

Code:

BSSM095T

Label:

Owner:

Number:

200

PIK constraint::

Source:

Options

in BSSM095S
{
index in BSSM0950
}

Column List

Name

CodeTypePM

ID USER
ID_USERINTEGERYesYes
ID SESSION

ID_SESSIONCHAR(16)Yes Yes
TMST CREATED

TMST_CREATEDTIMESTAMPNo Yes

BSSM096T
Name:
REGION
Code:

BSSM096T
Label:

RegionTable
Owner:
Number:

10
PK constraint:

BSSM0960
Source:

Options
in BSSM00CS
{
index in BSSM00C0
}

Description

This is a code type table that contains the regions in the state and the Central Office.

Column List

Name	Code	Type	P	M
CODE REGION	CODE_REGION	INTEGER	Yes	Yes
TEXT REGION	TEXT_REGION	CHAR(3)	No	Yes
CODE PRNT ROUTER	CODE_PRNT_ROUTER	CHAR(5)	No	Yes
TEXT DESC	TEXT_DESC	CHAR(40)	No	Yes

BSSM990T
Name:

CONV SKILL KEYWORD
Code:

BSSM990T

Label:
Owner:
Number:

18000
PIK constraint:
Source:

Options
in
BSSM990S
{
index in
BSSM9900
}

Description
This table will only be used during conversion. It provides a cross reference from old skill to new skill id.

Column List

Name	Code	Type	P	M
TEXT ODDS KEYWORD	TEXT_ODDS_KEYWORD	CHAR(5)	Yes	Yes
ID SKILL	ID_SKILL	INTEGER	Yes	Yes
ID SKILL TYPE	ID_SKILL_TYPE	INTEGER	No	Yes

A system according to the present invention has been made available through the World Wide Web with a URL of http://www.illinoisskillsmatch.com, all of which is incorporated by reference herein.

The method and system of the invention has been described with reference to a preferred embodiment suited for jobs; managing the submission of job related information; and matching job seekers to potential employers. It is to be understood that the method and system according to the invention is suitable for other applications involving the matching of groups or members of groups based on various criteria. Such applications include scholarships; group affiliations and memberships; intra-company tasks and assignments; and food service.

While the invention has been described and shown in connection with the preferred embodiment, it is to be understood that modifications may be made without departing from the spirit thereof. The embodiment described is by way of example and should not be construed as limiting of the claims except where referenced to the specification is required for such construction. The claims set forth below are to define the scope of protection sought by this application.